ELETROGATE
APOSTILA KIT
ARDUINO
MAKER

Sumário

Parte I - Revisão de circuitos elétricos e componentes básicos

Introdução

A primeira parte da apostila faz uma revisão sobre circuitos elétricos, com destaque para questões práticas de montagem e segurança que surgem no dia-a-dia do usuário do Kit Arduino MAKER.

O conteúdo de circuitos elétricos aborda divisores de tensão e corrente, conversão de níveis lógicos, grandezas analógicas e digitais, níveis de tensão e cuidados práticos de montagem.
Em relação aos componentes básicos, é feita uma breve discussão sobre resistores elétricos, capacitores, leds, diodos, chaves e protoboards.

O Arduino UNO é o principal componente do Kit e é discutido e introduzido em uma seção à parte, na Parte II da apostila.

Todos os conteúdos da Parte I são focados na apostila Arduino MAKER, tendo em vista a utilização adequada dos componentes e da realização prática de montagens pelos usuários. No entanto, recomenda-se a leitura das referências indicadas ao final de cada seção para maior aprofundamento.

O leitor veterano, já acostumado e conhecedor dos conceitos essenciais de eletrônica e eletricidade, pode pular a Parte I e ir direto a Parte II, na qual são apresentadas uma seção de exemplo de montagem para cada sensor ou componente importante da apostila.

Algum conhecimento prévio é bem-vindo, mas não é necessário para utilização do kit. Caso seja seu primeiro contato, nós recomendamos que antes de fazer as montagens você leia esses três artigos do http://blog.eletrogate.com/

- https://blog.eletrogate.com/o-que-e-arduino-para-que-serve-vantagens-e-como-utilizar/
- *http://blog.eletrogate.com/programacao-arduino-parte-1/*
- *http://blog.eletrogate.com/programacao-arduino-parte-2/*

Preparado? Vamos começar!

Revisão de circuitos elétricos

A apostila Arduino MAKER, bem como todas as outras apostilas que tratam de Arduino e eletrônica em geral, tem como conhecimento de base as teorias de circuitos elétricos e de eletrônica analógica e digital.

Do ponto de vista da teoria de circuitos elétricos, é importante conhecer os conceitos de grandezas elétricas: Tensão, corrente, carga, energia potência elétrica. Em todos os textos sobre Arduino ou qualquer assunto que envolva eletrônica, você sempre terá que lidar com esses termos. Para o leitor que se inicia nessa seara, recomendamos desde já que mesmo que a eletrônica não seja sua área de formação, que conheça esses conceitos básicos.

Vamos começar pela definição de "circuito elétrico". ***Um circuito elétrico/eletrônico é uma interconexão de elementos elétricos/eletrônicos***. Essa interconexão pode ser feita para atender a uma determinada tarefa, como acender uma lâmpada, acionar um motor, dissipar calor em uma resistência e tantos outras.

O circuito pode estar **energizado** ou **desenergizado**. Quando está energizado, é quando uma fonte de tensão externa ou interna está ligada aos componentes do circuito. Nesse caso, uma corrente elétrica fluirá entre os condutores do circuito. Quando está desenergizado, a fonte de tensão não está conectada e não há corrente elétrica fluindo entre os condutores.

Mas atenção, alguns elementos básicos de circuitos, como os capacitores ou massas metálicas, são elementos que armazenam energia elétrica. Em alguns casos, mesmo não havendo fonte de tensão conectada a um circuito, pode ser que um elemento que tenha energia armazenada descarregue essa energia dando origem a uma corrente elétrica transitória no circuito. Evitar que elementos do circuito fiquem energizados mesmo sem uma fonte de tensão, o que pode provocar descargas elétricas posteriores (e em alguns casos, danificar o circuito ou causar choques elétricos) é um dos motivos dos sistemas de aterramento em equipamentos como osciloscópios e em instalações residenciais, por exemplo.

Em todo circuito você vai ouvir falar das grandezas elétricas principais, assim, vamos aprender o que é cada uma delas.

Carga e corrente elétrica

A grandeza mais básica nos circuitos elétricos é a carga elétrica. Carga é a propriedade elétrica das partículas atômicas que compõem a matéria, e é medida em Coulombs. Sabemos da física elementar que a matéria é constituída de elétrons, prótons e neutros. **A carga elementar é a carga de 1 elétron, que é igual a 1,602 x10-19 C**.

Do conceito de carga elétrica advém o **conceito de corrente elétrica, que nada mais é do que a taxa de variação da carga em relação ao tempo**, ou seja, quando você tem um fluxo de carga em um condutor, a quantidade de carga (Coulomb) que atravessa esse condutor por unidade de tempo, é chamada de corrente elétrica. A medida utilizada para corrente é o **Ampére(A).**

Aqui temos que fazer uma distinção importante. Existem corrente elétrica contínua e alternada:

- **Corrente elétrica contínua:** É uma corrente que permanece constante e em uma única direção durante todo o tempo.
- **Corrente elétrica alternada**: É uma corrente que varia senoidalmente (ou de outra forma) com o tempo.

Com o Arduino UNO, lidamos como correntes elétricas contínuas, pois elas fluem sempre em uma mesma direção. É diferente da corrente e tensão elétrica da tomada de sua casa, que são alternadas.

Ou seja, os seus circuitos com Arduino UNO sempre serão alimentados com grandezas contínuas (corrente e tensão contínuas).

Tensão elétrica

Para que haja corrente elétrica em um condutor, é preciso que os elétrons se movimentem por ele em uma determinada direção, ou seja, é necessário “alguém” para transferir energia para as cargas elétricas para movê-las. Isso é feito por uma força chamada **força eletromotriz (fem),** tipicamente representada por uma bateria. Outros dois nomes comuns para força eletromotriz são **tensão elétrica** e **diferença de potencial.**

O mais comum é você ver apenas "*tensão*" nos artigos e exemplos com Arduino. Assim, definindo formalmente o conceito: Tensão elétrica é a energia necessária para mover uma unidade de carga através de um elemento, e é medida em **Volts (V).**

Potência e energia

A tensão e a corrente elétrica são duas grandezas básicas, e juntamente com a potência e energia, são as grandezas que descrevem qualquer circuito elétrico ou eletrônico. A potência é definida como a variação de energia (que pode estar sendo liberada ou consumida) em função do tempo, e é medida em **Watts (W).** A potência está associada ao calor que um componente está dissipando e a energia que ele consume.

Nós sabemos da vida prática que uma lâmpada de 100W consome mais energia do que uma de 60 W. Ou seja, se ambas estiverem ligadas por 1 hora por exemplo, a lâmpada de 100W vai implicar numa conta de energia mais cara.
A potência se relaciona com a tensão e corrente pela seguinte equação:

$$P = V \times I$$

Essa é a chamada potência instantânea. Com essa equação você saber qual a potência dissipada em um resistor por exemplo, bastando que saiba a tensão nos terminais do resistor e a corrente que flui por ele. O conceito de potência é importante pois muitos hobbystas acabam não tendo noção de quais valores de resistência usar, ou mesmo saber especificar componentes de forma adequada.

Um resistor de 33 ohms de potência igual a 1/4W, por exemplo, não pode ser ligado diretamente em circuito de 5V, pois nesse caso a potência dissipada nele seria maior que a que ele pode suportar.

Vamos voltar a esse assunto em breve, por ora, tenha em mente que é importante ter uma noção da potência dissipada ou consumida pelos elementos de um circuito que você vá montar.

Por fim, a energia elétrica é o somatório da potência elétrica durante todo o tempo em que o circuito esteve em funcionamento. A energia é dada em **Joules (J)** ou **Wh (watt-hora).** A unidade Wh é interessante pois mostra que a energia é calculada multiplicando-se a potência pelo tempo (apenas para os casos em que a potência é constante).

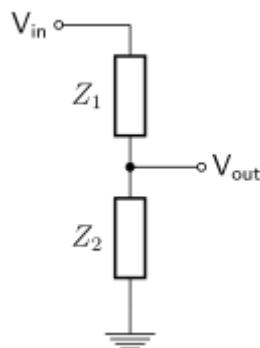
Essa primeira parte é um pouco conceitual, mas é importante saber de onde vieram toda a terminologia que você sempre vai ler nos manuais e artigos na internet. Na próxima seção, vamos discutir os componentes básicos de circuito que compõem o Kit Arduino MAKER.

Conversão de níveis lógicos e alimentação correta de circuitos

É muito comum que hobbystas e projetistas em geral acabem por cometer alguns erros de vez em quando. Na verdade, mesmo alguns artigos na internet e montagens amplamente usadas muitas vezes acabam por não utilizar as melhores práticas de forma rigorosa. Isso acontece muito no caso dos níveis lógicos dos sinais usados para interfacear o Arduino com outros circuitos.

Como veremos na seção de apresentação do Arduino UNO, o mesmo é alimentado por um cabo USB ou uma fonte externa entre 6V e 12V. O Circuito do Arduino possui reguladores de tensão que convertem a alimentação de entrada para 5V e para 3V. Os sinais lógicos das portas de saída(I/Os) do Arduino, variam entre 0 e 5V.

Isso significa que quando você quiser usar o seu Arduino UNO com um sensor ou CI que trabalhe com 3.3V, é preciso fazer a adequação dos níveis de tensão, pois se você enviar um sinal de 5V (saída do Arduino) em um circuito de 3.3V (CI ou sensor), você poderá queimar o pino daquele componente.
Em geral, sempre que dois circuitos que trabalhem com níveis de tensão diferentes forem conectados, é preciso fazer a conversão dos níveis lógicos. O mais comum é ter que abaixar saídas de 5V para 3.3V. Subir os sinais de 3.3V para 5V na maioria das vezes não é necessário pois o Arduino entende 3.3V como nível lógico alto, isto é, equivalente a 5V.

Para fazer a conversão de níveis lógicos você tem duas opções:

- Usar um divisor de tensão;
- Usar um CI conversor de níveis lógicos;

O divisor de tensão é a solução mais simples, mas usar um CI conversor é mais elegante e é o ideal. O divisor de tensão consiste em dois resistores ligados em série (Z1 e Z2) , em que o sinal de 5V é aplicado a o terminal de um deles. O terminal do segundo resistor é ligado ao GND, e o ponto de conexão entre os dois resistores é a saída do divisor, cuja tensão é dada pela seguinte relação:

$$V_{\text{out}} = \frac{Z_2}{Z_1 + Z_2} \cdot V_{\text{in}}$$

Em que Z1 e Z2 são os valores dos resistores da figura abaixo.

Um divisor de tensão muito comum é fazer Z1 igual 330 ohms e Z2 igual 680 ohms. Dessa forma a sáida Vout fica sendo 3.336 V. Como no Kit Arduino MAKER não há resistor de 680ohms, você pode ligar um de 330ohms como Z1 e dois de 330ohms como Z2. Os dois de 330 ligados em série formam um resistor de 660 ohm, o que resulta numa saída de 3.33V.

Vamos exemplificar como fazer um divisor de tensão como esse na seção de exemplos da parte II da apostila.

Revisão de componentes eletrônicos

O Kit Arduino MAKER possui os seguintes componentes básicos para montagens de circuitos:

- Buzzer Ativo 5V,
- LED Vermelho/ Verde/ Amarelo,
- Resistor 330Ω/ 1KΩ/ 10KΩ,
- Diodo 1N4007,
- Potenciômetro 10KΩ,
- Capacitor Cerâmico 10 nF/ 100 nF,
- Capacitor Eletrolítico 10uF/ 100uF,
- Chave Táctil (Push-button).

Vamos revisar a função de cada um deles dentro de um circuito eletrônica e apresentar mais algumas equações fundamentais para ter em mente ao fazer suas montagens.

Resistores elétricos

Os resistores são componentes que se opõem à passagem de corrente elétrica, ou seja, oferecem uma resistência elétrica. Dessa forma, quanto maior for o valor de um resistor, menor será a corrente elétrica que fluirá por ele e pelo condutor a ele conectada. A unidade de resistência elétrica é o Ohm(Ω), que é a unidade usada para especificar o valor dos resistores.

Os resistores mais comuns do mercado são construídos com fio de carbono e são vendidos em várias especificações. Os resistores do Kit são os tradicionais de 1/4W e 10% de tolerância. Isso significa que eles podem dissipar no máximo 1/4W (0,25 watts) e seu valor de resistência pode variar em até 10%. O resistor de 1KΩ pode então ter um valor mínimo de 900Ω e um valor máximo de 1100Ω.

Em algumas aplicações você pode precisar de resistores com precisão maior, como 5% ou 1%. Em outros casos, pode precisar de resistores com potência maior, como 1/2W ou menor, como 1/8W. Essas variações dependem da natureza de cada circuito.

Em geral, para as aplicações típicas e montagens de prototipagem que podem ser feitos com o Kit Arduino MAKER, os resistores tradicionais de 1/4W e 10% de tolerância são suficientes.

Outro ponto importante de se mencionar aqui é a Lei de Ohm, que relaciona as grandezas de tensão, corrente e resistência elétrica. A lei é dada por:

$$V = R \times I$$

Ou seja, se você sabe o valor de um resistor e a tensão aplicada nos seus terminais, você pode calcular a corrente elétrica que fluirá por ele. Juntamente com a equação para calcular potência elétrica, a lei de Ohm é importante para saber se os valores de corrente e potência que os resistores de seu circuito estão operando estão adequados.

Para fechar, você deve estar se perguntando, como saber o valor de resistência de um resistor? Você tem duas alternativas: Medir a resistência usando um multímetro ou determinar o valor por meio do código de cores do resistor.

Se você pegar um dos resistores do seu kit, verá que ele possui algumas faixas coloridas em seu corpo. Essas faixas são o código de cores do resistor. As duas primeiras faixas dizem os dois primeiros algarismos decimais. A terceira faixa colorida indica o multiplicador que devemos usar. E a última faixa, que fica um pouco mais afastada, indica a tolerância.

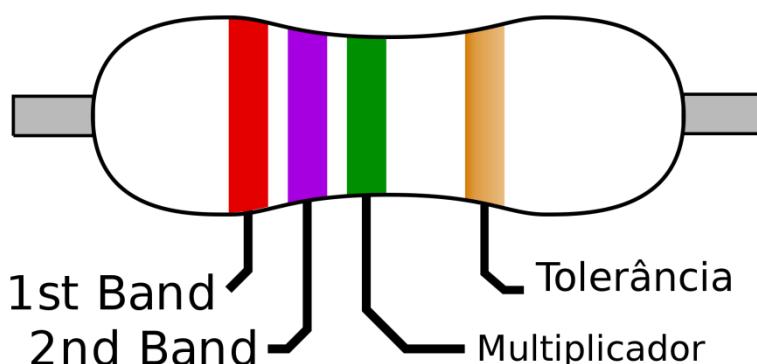



Figura 1: Faixas coloridas em um resistor

Na figura 2 apresentamos o código de cores para resistores. Cada cor está associada a um algarismo, um multiplicador e uma tolerância, conforme a tabela. Com a tabela você pode determinar a resistência de um resistor sem ter que usar o multímetro.

Mas atenção, fazer medições com o multímetro é recomendado principalmente se o componente já tiver sido utilizado, pois o mesmo pode ter sofrido algum dano ou mudança que não esteja visível.

Cor	Dígito	Multiplicador	Tolerância
Prata	-	x 0,01	± 10%
Dourado	-	x 0,1	± 5%
Preto	0	x 1	-
Marrom	1	x 10	± 1%
Vermelho	2	x 100	± 2%
Laranja	3	x 1K	-
Amarelo	4	x 10K	-
Verde	5	x 100K	± 0,5%
Azul	6	x 1M	± 0,25%
Violeta	7	x 10M	± 0,1%
Cinza	8	-	± 0,05%
Branco	9	-	-

Figura 2: Código de cores para resistores

Aplicando a tabela da figura 2 na imagem da figura 1, descobrimos que o resistor é de 2,7MΩ (Mega ohms) com tolerância de 5% (relativo à cor dourado da última tira).

Capacitores

Os capacitores são os elementos mais comuns nos circuitos eletrônicos depois dos resistores. São elementos que armazenam energia na forma de campos elétricos. Um capacitor é constituído de dois terminais condutores e um elemento dielétrico entre esses dois terminais, de forma que quando submetido a uma diferença de potencial, um campo elétrico surge entre esses terminais, causando o acúmulo de cargas positivas no terminal negativo e cargas negativas no terminal positivo.

São usados para implementar filtros, estabilizar sinais de tensão, na construção de fontes retificadores e várias outras aplicações.

O importante que você deve saber para utilizar o Kit é que os capacitores podem ser de quatro tipos:

- Eletrolíticos,
- Cerâmicos,
- Poliéster,
- Tântalo.

Capacitor eletrolítico

As diferenças de cada tipo de capacitor são a tecnologia construtiva e o material dielétrico utilizado. Capacitores eletrolíticos são feitos de duas longas camadas de alumínio (terminais) separadas por uma camada de óxido de alumínio (dielétrico). Devido a sua construção, eles possuem **polaridade,** o que significa que você obrigatoriamente deve ligar o terminal positivo (quem tem uma "perninha" maior) no polo positivo da tensão de alimentação, e o terminal negativo (discriminado no capacitor por uma faixa com símbolos de "-") obrigatoriamente no polo negativo da bateria. Do contrário, o capacitor será danificado.

Capacitores eletrolíticos costumam ser da ordem de micro Farad, sendo o **Farad** a unidade de medida de capacitância, usada diferenciar um capacitor do outro. A Figura 3 ilustra um típico capacitor eletrolítico.

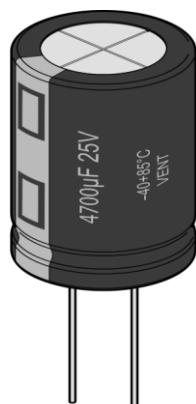



Figura 3: Capacitor eletrolítico 4700 micro Farads / 25 V

Capacitores cerâmicos

Capacitores cerâmicos não possuem polaridade, e são caracterizados por seu tamanho reduzido e por sua cor característica, um marrom claro um tanto fosco. Possuem capacitância da ordem de **pico Farad**. Veja nas imagens abaixo um típico capacitor cerâmico e sua identificação:

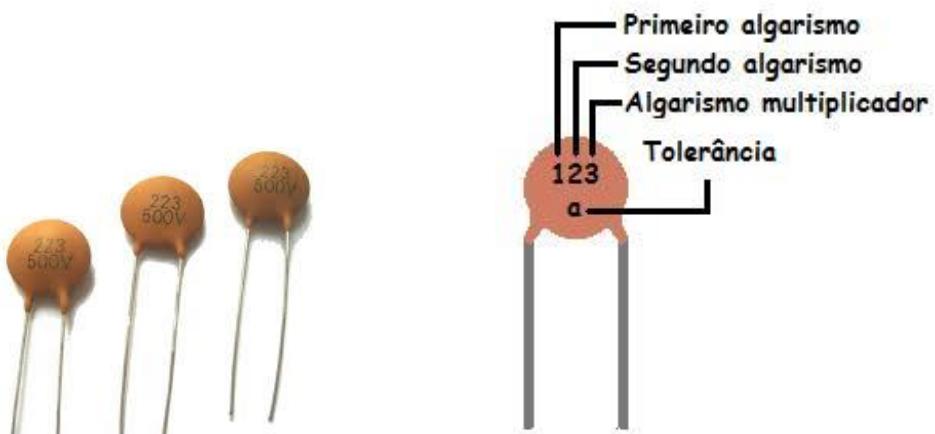



Figura 4: Capacitores cerâmicos

Na imagem da esquerda, os capacitores possuem o valor de **22 nano Farads** para a faixa de tensão de até 500V.

223 = 22 x 1000 = 22.000 pF = 22 nF

Por fim, há também os capacitores de poliéster e de tântalo. No Kit Arduino MAKER, você receberá apenas exemplares de capacitores eletrolíticos e cerâmicos.

Diodos e Leds

Diodos e Leds são tratados ao mesmo tempo pois tratam na verdade do mesmo componente. Diodos são elementos semicondutores que só permitem a passagem de corrente elétrica em uma direção.

São constituídos de dois terminais, o **Anodo (+)** e o **catodo (-)**, sendo que para que possa conduzir corrente elétrica, é preciso conectar o Anodo na parte positiva do circuito, e o Catodo na parte negativa. Do contrário, o diodo se comporta como um circuito aberto.

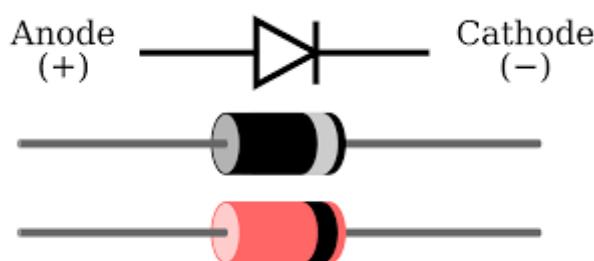



Figura 4: Diodo e seus terminais

Na figura 4, você pode ver que o componente possui uma faixa indicadora do terminal de catodo. O diodo do kit Arduino MAKER é um modelo tradicional e que está no mercado há muitos anos, o 1N4007.

O LED é um tipo específico de diodo - **Light Emitter Diode**, ou seja, diodo emissor de luz. Trata-se de um diodo que quando polarizado corretamente, emite luz para o ambiente externo. O Kit Arduino BEGINNIG vem acompanhado de leds na cor vermelha, verde e amarelo, as mais tradicionais.

Nesse ponto, é importante você saber que sempre deve ligar um led junto de um resistor, para que a corrente elétrica que flua pelo led não seja excessiva e acabe por queimá-lo. Além disso, lembre-se que por ser um diodo, o led só funciona se o Anodo estiver conectado ao polo positivo do sinal de tensão.

Para identificar o Anodo do Led, basta identificar a perna mais longa do componente, como na imagem abaixo:

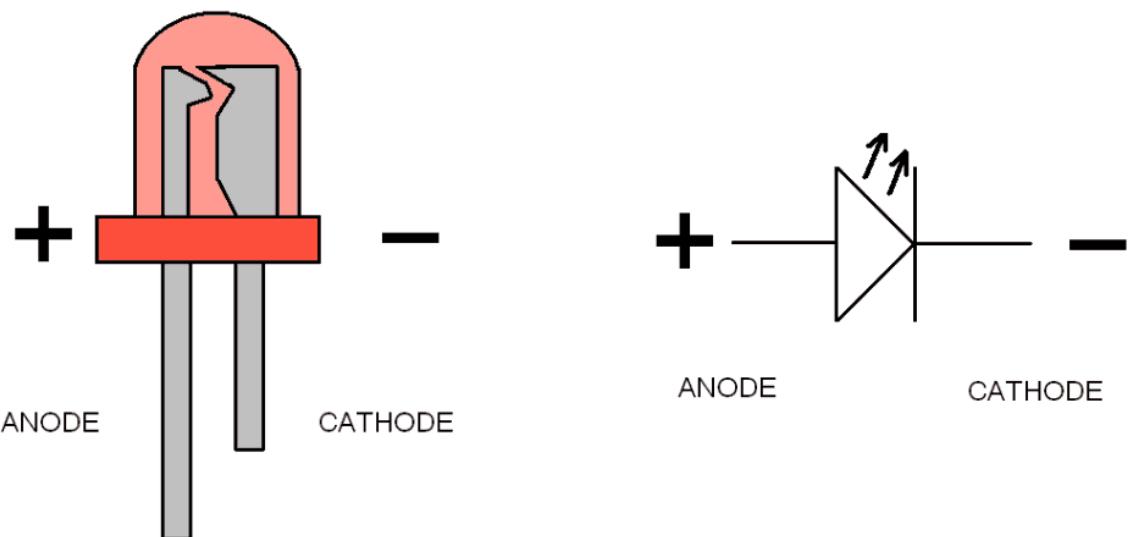
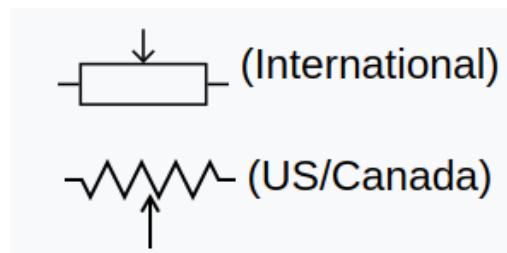



Figura 5: Terminais de um Led. Créditos: Build-eletronic-circuits.com

Os demais componentes da apostila, Buzzer, potenciômetro e push-buttons, serão explicados em seções de exemplos, nas quais iremos apresentar um circuito prático para montagem onde será explicado o funcionamento dos mesmos.

Os sensores do Kit Arduino MAKER, quais sejam: Sensor de Luz LDR e sensor de temperatura NTC, juntamente com o Micro Servo 9g SG90 TowerPro, também terão uma seção de exemplo dedicada a cada um deles.

Parte II - Seção de Exemplos Práticos

Agora vamos entrar nas seções de exemplos em si. Os conceitos da Parte I são importantes caso você esteja começando a trabalhar com eletrônica. No entanto, devido ao aspecto prático da montagem, não necessariamente você precisa de ler toda a parte introdutória para montar os circuitos abaixo.

Em cada exemplo vamos apresentar os materiais utilizados, o diagrama de circuito e o código base com as devidas explicações e sugestões de referências.

Exemplo 1 - Usando potenciômetro para fazer leituras analógicas

O potenciômetro nada mais do que um resistor cujo valor de resistência pode ser ajustado de forma manual. Existem potenciômetros slides e giratórios. Na figura abaixo mostramos um potenciômetro giratório dos mais comumente encontrados no mercado.

Em ambos, ao variar a posição da chave manual, seja ela giratória ou slide, o valor da resistência entre o terminal de saída e um dos outros terminais se altera. O símbolo do potenciômetro é mostrado na seguinte imagem:

Nesse exemplo, vamos ligar a saída de um potenciômetro a uma entrada analógica da Arduino UNO. Dessa forma, vamos ler o valor de tensão na saída do potenciômetro e vê-la variando de
0 a 1023. Mas como assim, 0 a 1023?

Isso se deve ao seguinte. Vamos aplicar uma tensão de 5V nos terminais do potenciômetro. A entrada analógica do Arduino, ao receber um sinal de tensão externo, faz a conversão do mesmo para um valor digital, que é representado por um número inteiro de 0 a 1023. Esses números são assim devido à quantidade de bits que o conversor analógico digital do Arduino trabalha, que são 10 bits (2 elevado a 10 = 1024).

Ou seja, o Arduino divide o valor de tensão de referência em 1024 unidades (0 a 1023) de 0,00488 volts. Assim, se a tensão lida na entrada analógica for de 2,5V, o valor capturado pelo Arduino será metade de 2,5/0,00488 = 512. Se for 0V, será 0, e ser for 5V, será 1023, e assim proporcionalmente para todos os valores. Assim, digamos que o valor de tensão se dado por V. O valor que o Arduino vai te mostrar é:

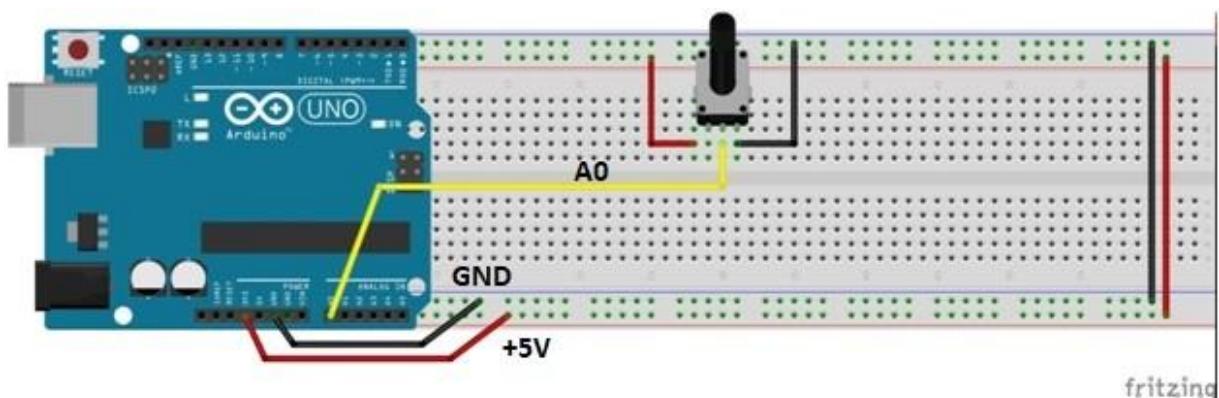
Valor = (V/5) * 1024

Em que 5V é o valor de referência configurado no conversor analógico-digital (uma configuração já padrão, não se preocupe com ela) e 1024 é igual 2 elevado a 10.

No nosso código, queremos saber o valor de tensão na saída do potenciômetro, e não esse número de 0 a 1023, assim, rearranjar a equação para o seguinte:

Tensão = Valor * (5/1024)

Bacana, né? Agora, vamos à montagem em si.

Lista de materiais:

Para esse exemplo você vai precisar:

- Arduino UNO;
- Protoboard;
- Potenciômetro 10K;
- Jumpers de ligação;

Diagrama do circuito

Monte o circuito conforme diagrama abaixo e carregue o código de exemplo:

O Potenciômetro possui 3 terminais, sendo que o do meio é o que possui resistência variável. A ligação consiste em ligar os dois terminais fixos a uma tensão de 5V. Assim, o terminal intermediário do potenciômetro terá um valor que varia de 0 a 5V à medida que você gira seu knob.

O terminal intermediário é ligado diretamente a uma entrada analógica do Arduino (A0). Como a tensão é de no máximo 5V, então não há problema em ligar direto.

Carregue o código abaixo no Arduino e você verá as leituras no Monitor serial da IDE Arduino.

```
// Exemplo 1 - Usando potenciometro para fazer leituras analógicas
// Apostila Eletrogate - KIT MAKER

#define sensorPin A0      // define entrada analógica A0

int sensorValue = 0;      // variável inteiro igual a zero
float voltage;            // variável numero fracionario

void setup()
{
  Serial.begin(9600);     // monitor serial - velocidade 9600 Bps
  delay(100);             // atraso de 100 milisegundos
}

void loop()
{
  sensorValue = analogRead(sensorPin);       // leitura da entrada analógica A0
  voltage = sensorValue * (5.0 / 1024);      // cálculo da tensão

  Serial.print("Tensão do potenciometro: "); // imprime no monitor serial
  Serial.print(voltage);                     // imprime a tensão
  Serial.print("    Valor: ");               // imprime no monitor serial
  Serial.println(sensorValue);               // imprime o valor
  delay(500);                                // atraso de 500 milisegundos
}
```

No código acima, nós declaramos a variável **sensorValue** para armazenar as leituras da entrada analógica A0 e a variável do tipo **float Voltage** para armazenar o valor lido convertido para tensão.

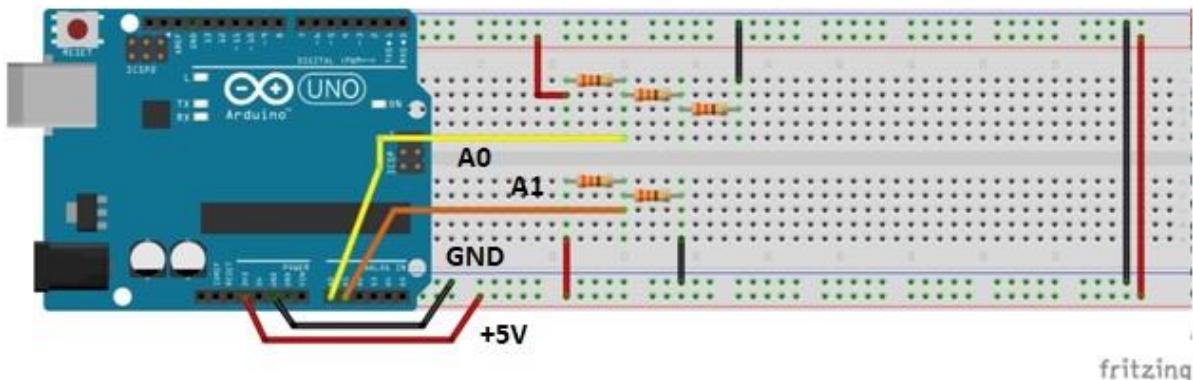
Na função **void Setup()**, nós inicializamos o terminal serial com uma taxa de transmissão de 9600 kbps. Na função **void Loop(),** primeiro faz-se a leitura da entrada analógica A0 com a função **analogRead(SensorPin)** e armazenamos a mesma na variável **sensorValue**. Em seguida, aplicamos a fórmula para converter a leitura (que é um número entre 0 e 1023) para o valor de tensão correspondente. O resultado é armazenado na variável **Voltage** e em seguido mostrado na interface serial da IDE Arduino.

Referência:

- https://www.arduino.cc/en/Tutorial/ReadAnalogVoltage

Exemplo 2 - Divisor de tensão

Esse exemplo é para ilustrar como usar um divisor de tensão. Sempre que você precisar abaixar um sinal lógico de 5V para 3.3V você pode usar esse circuito. Explicamos o divisor de tensão na seção de introdução, mais especificamente, quando conversamos sobre conversão de níveis lógicos.

Esse circuito será útil sempre que você tiver que abaixar as saídas do Arduino de 5V para 3.3V.

Lista de materiais:

Para esse exemplo você vai precisar:

- 3 resistores de 330 ohms;
- 2 resistores de 1K ohm;
- 1 Arduino UNO;
- Protoboard;
- Jumpers de ligação;

Diagrama do circuito:

Esse é um diagrama com dois divisores de tensão. O primeiro é composto por três resistores, sendo cada um de 330. Assim, o resistor Z1 é de 330 e o resistor Z2 é a associação série dos outros dois, resultando numa resistência de 660. Dessa forma, a tensão de saída do divisor é:

$$(660 / 330 + 660) * 5 = 3{,}33V$$

O segundo divisor é formado por dois resistores de 1K, dessa forma, a tensão de saída é a tensão de entrada dividida pela metade:

$$(1000 / 1000 + 1000) * 5 = 2{,}5 V$$

O código para o exemplo 2 é uma extensão do código usada na seção anterior para ler valores de tensão do potenciômetro. Nesse caso, nós fazemos a leitura de dois canais analógicos (A0 e A1), fazemos as conversões para as tensões e mostramos os resultados de cada divisor na interface serial.

Referências:

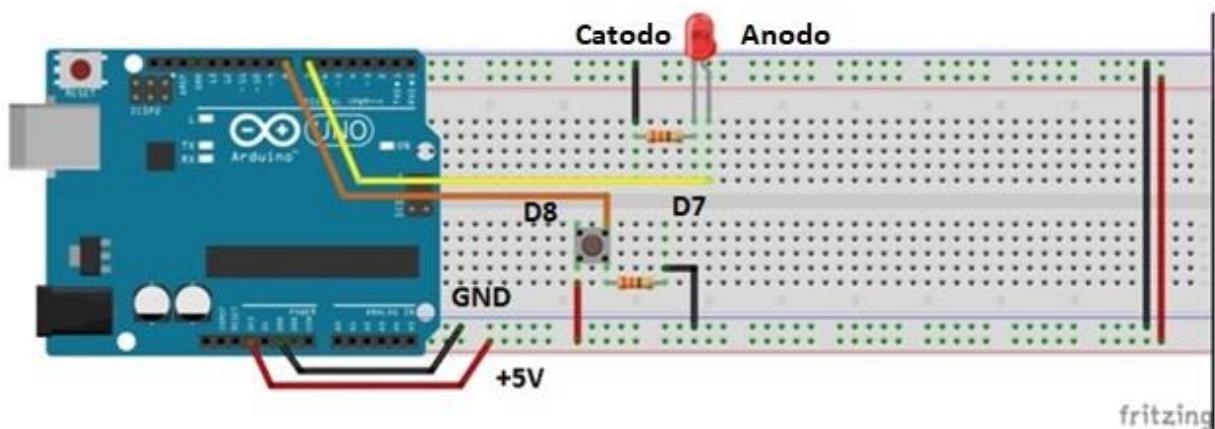
- https://www.arduino.cc/en/Tutorial/ReadAnalogVoltage
- http://www.ufjf.br/fisica/files/2013/10/FIII-05-07-Divisor-de-tens%C3%A3o.pdf
- http://www.sofisica.com.br/conteudos/Eletromagnetismo/Eletrodinamica/associacaoderesistores.php

Carregue o código abaixo e observe os dois valores no Monitor Serial da IDE Arduino.

```
// Exemplo 2 - Divisor de tensão
// Apostila Eletrogate - KIT MAKER

#define sensorPin1 A0            // define entrada analógica A0
#define sensorPin2 A1            // define entrada analógica A1

int sensorValue1 = 0;            // variavel inteiro igual a zero
int sensorValue2 = 0;            // variavel inteiro igual a zero
float voltage1;                  // variavel numero fracionario
float voltage2;                  // variavel numero fracionario

void setup()
{
  Serial.begin(9600);            // monitor serial - velocidade 9600 Bps
  delay(100);                    // atraso de 100 milisegundos
}

void loop()
{
  sensorValue1 = analogRead(sensorPin1);   // leitura da entrada analógica A0
  sensorValue2 = analogRead(sensorPin2);   // leitura da entrada analógica A1
  voltage1 = sensorValue1 * (5.0 / 1024);  // cálculo da tensão 1
  voltage2 = sensorValue2 * (5.0 / 1024);  // cálculo da tensão 2
  Serial.print("Tensao do divisor 1: ");   // imprime no monitor serial
  Serial.print(voltage1);                  // imprime a tensão 1
  Serial.print("  Tensao do divisor 2: "); // imprime no monitor serial
  Serial.println(voltage2);                // imprime a tensão 2
  delay(500);                              // atraso de 500 milisegundos
}
```

Exemplo 3 - Entradas e saídas digitais - push-button + led

Os push-buttons (chaves botão) e leds são elementos presentes em praticamente qualquer circuito eletrônico. As chaves são usadas para enviar comandos para o Arduino e os Leds são elementos de sinalização luminosa.

Esses dois componentes são trabalhados por meio das entradas e saídas digitais do Arduino. Neste exemplo vamos fazer uma aplicação básica que você provavelmente vai repetir muitas vezes. Vamos ler o estado de um push-button e usá-la para acender ou apagar um led. Ou seja, sempre que o botão for acionado, vamos apagar ou acender o Led.

Lista de materiais:

Para esse exemplo você vai precisar:

- 2 resistores de 330 ohms;
- 1 Led vermelho (ou de outra cor de sua preferência);
- Push-button (chave botão);
- 1 Arduino UNO;
- Protoboard;
- Jumpers de ligação;

Diagrama do circuito:

Esse pequeno código abaixo mostra como ler entradas digitais e com acionar as saídas digitais. Na função **void Setup(),** é preciso configurar qual pino será usado como saída e qual será usado como entrada.

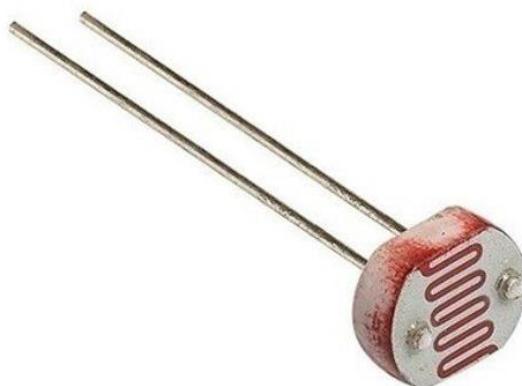
Depois de configurar os pinos, para acioná-los basta chamar a função **digitalWrite(pino,HIGH).** A função **digitalWrite()** aciona ou desaciona um pino digital dependendo do valor passado no argumento. Se for “HIGH”, o pino é acionado. Se for “LOW”, o pino é desligado.

Na função **void Loop(),** fizemos um if no qual a função **digitalRead** é usada para saber se o pushButton está acionado ou não. Caso ele esteja acionado, nós acendemos o Led, caso ele esteja desligado, nós desligamos o led.

Carregue o código abaixo e pressione o botão para acender o LED.

```
// Exemplo 3 - Entradas e saídas digitais - push-button + led
// Apostila Eletrogate - KIT MAKER

#define PinButton 8                  // define pino digital D8
#define ledPin 7                     // define pino digital D7

void setup()
{
  pinMode(PinButton, INPUT);         // configura D8 como entrada digital
  pinMode(ledPin, OUTPUT);           // configura D7 como saída digital
  Serial.begin(9600);                // monitor serial - velocidade 9600 Bps
  delay(100);                        // atraso de 100 milisegundos
}

void loop()
{
  if ( digitalRead(PinButton) == HIGH)  // se chave = nivel alto
  {
    digitalWrite(ledPin, HIGH);         // liga LED com 5V
    Serial.print("Acendendo Led");      // imprime no monitor serial
  }
  else                                  // senão chave = nivel baixo
  {
    digitalWrite(ledPin, LOW);          // desliga LED com 0V
    Serial.print("Desligando led");     // imprime no monitor serial
  }
  delay(100);                           // atraso de 100 milisegundos
}
```

Referências:

- https://www.arduino.cc/reference/en/language/functions/digital-io/digitalread/
- https://www.arduino.cc/reference/en/language/functions/digital-io/digitalwrite/

Exemplo 4 - Sensor de luz LDR

O sensor LDR é um sensor de luminosidade. LDR é um **Light Dependent Resistor**, ou seja, um resistor cuja resistência varia com a quantidade de luz que incide sobre ele. Esse é seu princípio de funcionamento.

Quanto maior a luminosidade em um ambiente, menor a resistência do LDR. Essa variação na resistência é medida através da queda de tensão no sensor, que varia proporcionalmente (de acordo com a lei Ohm, lembra?) com a queda na resistência elétrica.

A imagem abaixo mostra o sensor em mais detalhes:

Fotoresistor (LDR)

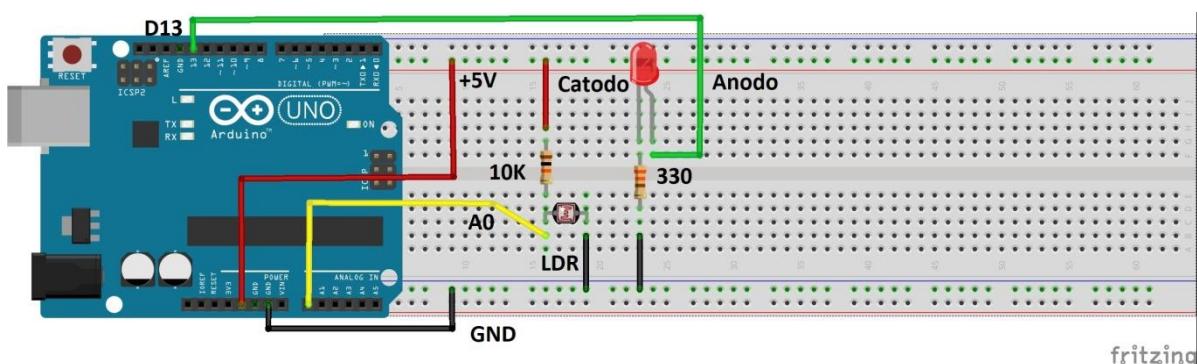
É importante considerar a potência máxima do sensor, que é de 100 mW. Ou seja, com uma tensão de operação de 5V, a corrente máxima que pode passar por ele é 20 mA. Felizmente, com 8K ohms (que medimos experimentalmente com o ambiente bem iluminado), que é a resistência mínima, a corrente ainda está longe disso, sendo 0,625mA. Dessa forma, podemos interfacear o sensor diretamente com o Arduino.

**Nota: Nas suas medições, pode ser que você encontre um valor de resistência mínimo diferente, pois depende da iluminação local.*

Especificações do LDR:

- Modelo: GL5528
- Diâmetro: 5mm
- Tensão máxima: 150VDC
- Potência máxima: 100mW
- Temperatura de operação: -30°C a 70°C
- Comprimento com terminais: 32mm
- Resistência no escuro: 1 MΩ (Lux 0)
- Resistência na luz: 10-20 KΩ (Lux 10)

Este sensor de luminosidade pode ser utilizado em projetos com Arduino e outros microcontroladores para alarmes, automação residencial, sensores de presença e vários outros.

Nesse exemplo, vamos usar uma entrada analógica do Arduino para ler a variação de tensão no LDR e, consequentemente, saber como a luminosidade ambiente está se comportando. Veja na especificação que com muita luz, a resistência fica em torno de 10-20 KΩ, enquanto no escuro pode chegar a 1MΩ.

Para podermos ler as variações de tensão resultantes da variação da resistência do LDR, vamos usar o sensor como parte de um divisor de tensão. Assim, a saída do divisor será dependente apenas da resistência do sensor, pois a tensão de entrada e a outra resistência são valores conhecidos. No nosso caso, vamos usar um resistor de 10K e uma tensão de operação de 5V. Assim, o sinal que vamos ler no Arduino terá uma variação de 2,2V (quando o LDR for 8K) e 5V (quando o LDR tiver resistências muito maiores que o resistor de 10K).

Lista de materiais:

Para esse exemplo você vai precisar:

- LDR;
- Resistor de 10k;
- 1 Arduino UNO;
- Protoboard;
- Jumpers de ligação;

Diagrama do circuito:

No diagrama, o sensor é ligado como parte de um divisor de tensão no pino analógico A0, de forma que a tensão de saída do divisor varia de acordo com a variação da resistência do sensor. Assim, vamos identificar as variações na intensidade de luz pelas variações na tensão do sensor.

Quanto maior a intensidade de luz, menor a resistência do sensor e, consequentemente, menor a tensão de saída.

Carregue o código abaixo e varie a luminosidade sobre o LDR.

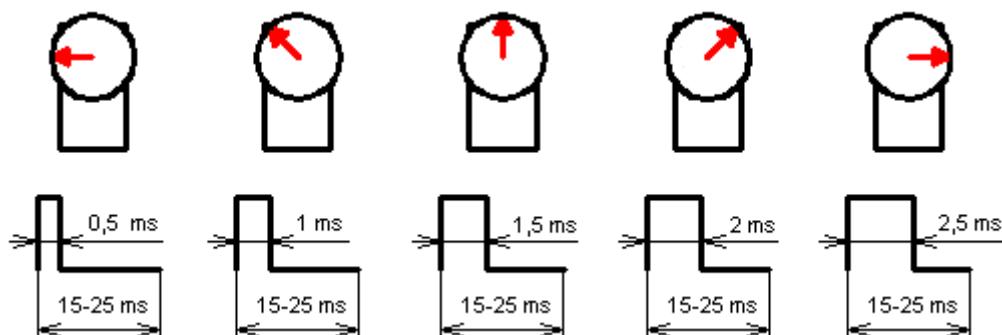
```
// Exemplo 4 - Sensor de luz LDR
// Apostila Eletrogate - KIT MAKER

#define AnalogLDR A0                    // define pino analógico A0
#define Limiar 1.5                      // define constante igual a 1.5
#define ledPin 13                       // define pino digital D13

int Leitura = 0;                        // variavel inteiro igual a zero
float VoltageLDR;                       // variavel numero fracionario
float ResLDR;                           // variavel numero fracionario

void setup()
{
  pinMode(ledPin, OUTPUT);                  // configura D13 como saída digital
  Serial.begin(9600);                       // monitor serial - velocidade 9600 Bps
  delay(100);                               // atraso de 100 milisegundos
}

void loop()
{
  Leitura = analogRead(AnalogLDR);          // leitura da tensão no pino analogico A0
  VoltageLDR =  Leitura * (5.0/1024);       // calculo da tensão no LDR
  Serial.print("Leitura sensor LDR = ");    // imprime no monitor serial
  Serial.println(VoltageLDR);               // imprime a tensão do LDR

  if (VoltageLDR > Limiar)                  // se a tensão LDR maior do que limiar
    digitalWrite(ledPin, HIGH);             // liga LED com 5V
  else                                      // senão a tensão LDR < limiar
    digitalWrite(ledPin, LOW);              // desliga LED com 0V
  delay(500);                               // atraso de 500 milisegundos
}
```

No código acima usamos funcionalidades de todos os exemplos anteriores. Como o sensor é um elemento de um divisor de tensão, fazemos a sua leitura do mesmo modo que nos exemplos do potenciômetro e divisor de tensão.

Nesse caso, definimos uma tensão de limiar, a partir da qual desligamos ou ligamos um led para indicar que a intensidade da luz ultrapassou determinado valor. Esse limiar pode ser ajustado por você para desligar o led em intensidades diferentes de luz ambiente.

É importante que o sensor esteja exposto à iluminação ambiente e não sofra interferência de fontes luminosas próximas, mas que não sejam parte do ambiente.

Referências:

- https://maker.pro/education/using-an-ldr-sensor-with-arduino-a-tutorial-for-beginners
- http://blog.eletrogate.com/controle-de-luminosidade-com-arduino-e-sensor-ldr/

Exemplo 5 - Acionando o Micro Servo 9g SG90 TowerPro

Um servomotor é um equipamento eletromecânico que possui um encoder e um controlador acoplado. Diferentemente de motores tradicionais, como de corrente contínua, o servo motor apresenta movimento rotativo proporcional a um comando de forma a atualizar sua posição. Ao invés de girar continuamente como os motores de corrente contínua, o servo ao receber um comando, gira até a posição especificada pelo mesmo.

Ou seja, o servo motor é um atuador rotativo para controle de posição, que atua com precisão e velocidade controlada em malha fechada. De acordo com a largura do pulso aplicado no pino de controle PWM, a posição do rotor é definida (0 a 180 graus). Os pulsos devem variar entre 0,5 ms e 2,5 ms.

Posição do Servo de acordo com a largura do pulso
Fonte: electronics.stackexchange.com

Existem dois tipos de Servomotores:

- Servomotor analógico
- Servomotor digital

Servomotores analógicos são os mais comuns e mais baratos. O controle da posição do rotor utiliza um método analógico através da leitura de tensão sobre um potenciômetro interno.

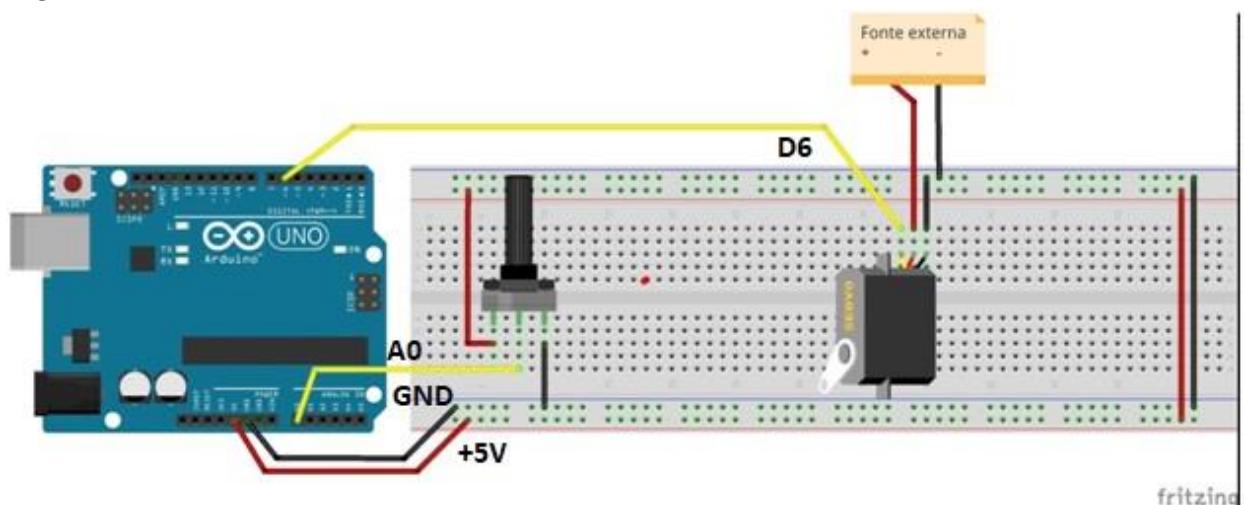
No caso dos servomotores digitais, mais caros e mais precisos, o controle da posição do rotor utiliza um encoder digital.

A figura abaixo mostra um típico **servo motor analógico**. Trata-se do **Micro Servo Tower Pro SG90 9G** que acompanha o kit Arduino MAKER.

Os Servos são acionados por meio de três fios, dois para alimentação e um correspondente ao sinal de controle para determinar a posição. Em geral, os três fios são:

- Marrom: GND,
- Vermelho: Alimentação positiva,
- Laranja: Sinal de controle PWM.

Lista de materiais:

Antes de mais nada, vamos separar os componentes que precisamos para montar o servomotor junto com o Arduino. A nossa lista de componentes é a seguinte:

- Micro Servo 9g SG90 TowerPro;
- Arduino UNO + cabo USB;
- Potenciômetro de 10k;
- Fonte 5V de protoboard para alimentar o servo;
- Jumpers para conexão no protoboard;
- Push button;

A montagem é simples. O servomotor em si deve ser ligado à alimentação conforme as cores apresentadas na introdução. De acordo com as especificações de tensão e corrente dos pinos do Arduino UNO, os pinos de VCC e GND da placa conseguem fornecer até 200 mA. Ao utilizar servomotores, é recomendado que você utilize uma fonte externa, como a fonte para protoboard que inserimos na lista de materiais.

Diagrama do circuito:

A montagem para uma aplicação de controle do servo em qualquer posição, fica da seguinte forma:

Carregue o código abaixo e gire o potenciômetro para o Servo motor girar:

```
// Exemplo 5 - Acionando o Micro Servo TowerPro
// Apostila Eletrogate - KIT MAKER

#include <Servo.h>                       // usando bilbioteca Servo

Servo myservo;                           // cria o objeto myservo

#define potpin A0                        // define pino analógico A0

int val;                                 // variavel inteiro

void setup()
{
  myservo.attach(6);                     // configura pino D6 - controle do Servo
}

void loop()
{
  val = analogRead(potpin);              // leitura da tensão no pino A0
  val = map(val, 0, 1023, 0, 179);       // converte a leitura em números (0-179)
  myservo.write(val);                    // controle PWM do servo
  delay(15);                             // atraso de 15 milisegundos
}
```

O aspecto mais importante desse software é a utilização da biblioteca **servo.h**. Esta biblioteca possui as funções necessárias para posicionar a servo para a posição desejada. Na função **Void Setup()** nós associamos o objeto servomotor, do tipo Servo, a um pino do arduino. Na mesma função, nós inicializamos o servo na posição 0º, utilizando o método **write** do objeto que acabamos de criar.

Na função **Void Loop(),** o procedimento consiste em ler o valor do potenciômetro e usá-lo como referência para atualizar a posição do servo. A leitura analógica do potenciômetro retorna um valor entre 0 e 1023. Para controlar o servo nós usamos valores de 0 a 179, correspondendo ao meio giro de 180º do mesmo. Assim, é necessário usar a função **Map()** para traduzir a escala de 0-1023 para a escala de 0-179. Dessa forma, os valores lidos do potenciômetro podem ser usados como referência para determinar a posição do servomotor.

Para atualizar a posição do servo a cada iteração do loop, nós chamamos o método **write()** do objeto servomotor, passando como parâmetro o valor lido do potenciômetro traduzido para a escala de 0-179. Assim, sempre que mexermos no potenciômetro, o servo motor irá atuar e atualizar a sua posição.

Referências:

- http://blog.eletrogate.com/servo-motor-para-aplicacoes-com-arduino/
- http://blog.eletrogate.com/kit-braco-robotico-mdf-com-arduino/
- https://www.arduino.cc/en/Tutorial/Knob
- https://www.allaboutcircuits.com/projects/servo-motor-control-with-an-arduino/
- http://www.instructables.com/id/Arduino-Servo-Motors/

Exemplo 6 - Sensor de temperatura NTC

O sensor de temperatura NTC pertence a uma classe de sensores chamada de Termistores. São componentes cuja resistência é dependente da temperatura. Para cada valor de temperatura absoluta há um valor de resistência.

Assim, o princípio para medir o sinal de um termistor é o mesmo que usamos para medir o sinal do LDR. Vamos medir na verdade, a queda de tensão provocada na resistência do sensor.

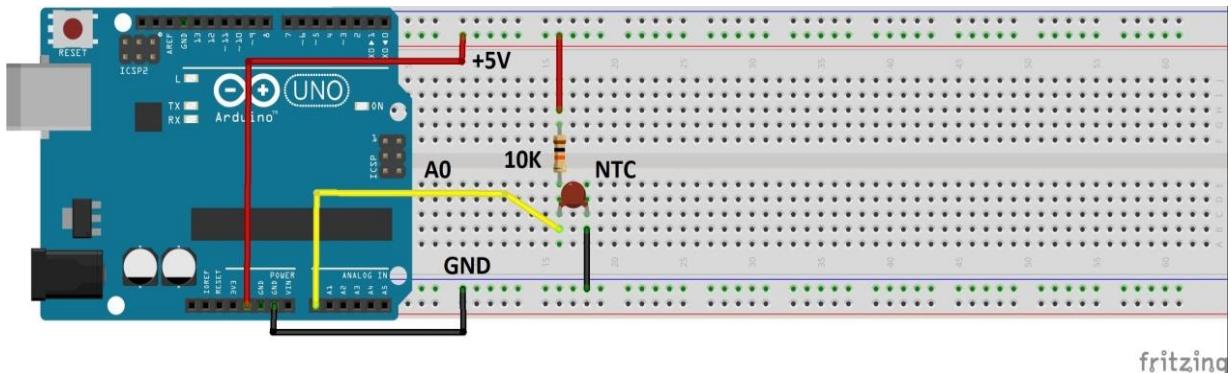
Existem dois tipos básicos de termistores:

- **PTC (Positive temperature coeficient):** Nesse sensor, a resistência elétrica aumenta à medida que a temperatura aumenta;

- **NTC (Negative Temperature Coeficient):** Nesse sensor, a resistência elétrica diminui à medida que a temperatura aumenta.

O termistor que acompanha o kit é do tipo NTC, como o próprio nome diz. Esse é o tipo mais comum do mercado.

O grande problema dos termistores é que é necessária fazer uma função de calibração, pois a relação entre temperatura e resistência elétrica não é linear. Existe uma equação, chamada equação de **Steinhart-Hart** que é usada para descrever essa relação (vide referências).

O código que vamos usar nesse exemplo utiliza a equação de Steinhart-Hart conforme a referência 1.

Lista de materiais:

Antes de mais nada, vamos separar os componentes que precisamos para montar o NTC junto com o Arduino. A nossa lista de componentes é a seguinte:

- 1 Sensor de temperatura NTC;
- Arduino UNO + cabo USB;
- 1 resistor de 10k;
- Protoboard;
- Jumpers para conexão no protoboard;

Diagrama do circuito:

Para uma melhor precisão nas medidas de temperatura, meça a tensão de 5V no barramento do protoboard, usando um multímetro (não contém no KIT). Essa tensão é a mesma usada como referência do conversor analógico-digital do Arduino. Especifique esse valor de tensão na primeira linha do Sketch:

```
#define Vin 5.0
```

Por exemplo, se a tensão for 4,85 V:

```
#define Vin 4.85
```

Carregue o código abaixo e meça a temperatura com o NTC:

```
// Exemplo 6 - Sensor de temperatura NTC
// Apostila Eletrogate - KIT MAKER

#define Vin 5.0                  // define constante igual a 5.0
#define T0 298.15                // define constante igual a 298.15 Kelvin
#define Rt 10000                 // Resistor do divisor de tensao
#define R0 10000                 // Valor da resistencia inicial do NTC
#define T1 273.15                // [K] in datasheet 0º C
#define T2 373.15                // [K] in datasheet 100° C
#define RT1 35563                // [ohms] resistencia in T1
#define RT2 549                  // [ohms] resistencia in T2
float beta = 0.0;                // parametros iniciais [K]
float Rinf = 0.0;                // parametros iniciais [ohm]
float TempKelvin = 0.0;          // variable output
float TempCelsius = 0.0;         // variable output
float Vout = 0.0;                // Vout in A0
float Rout = 0.0;                // Rout in A0

void setup()
{
  Serial.begin(9600);                     // monitor serial - velocidade 9600 Bps
  beta = (log(RT1 / RT2)) / ((1 / T1) - (1 / T2));      // calculo de beta
  Rinf = R0 * exp(-beta / T0);            // calculo de Rinf
  delay(100);                             // atraso de 100 milisegundos
}

void loop()
{
  Vout = Vin * ((float)(analogRead(0)) / 1024.0); // calculo de V0 e leitura de A0
  Rout = (Rt * Vout / (Vin - Vout));             // calculo de Rout
  TempKelvin = (beta / log(Rout / Rinf));        // calculo da temp. em Kelvins
  TempCelsius = TempKelvin - 273.15;             // calculo da temp. em Celsius
  Serial.print("Temperatura em Celsius: ");      // imprime no monitor serial
  Serial.print(TempCelsius);                     // imprime temperatura Celsius
  Serial.print("   Temperatura em Kelvin: ");    // imprime no monitor serial
  Serial.println(TempKelvin);                    // imprime temperatura Kelvins
  delay(500);                                    // atraso de 500 milisegundos
}
```

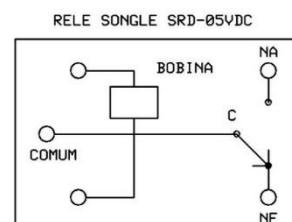
Referências e sugestões de leitura:

- http://www.instructables.com/id/NTC-Temperature-Sensor-With-Arduino/
- https://www.ametherm.com/thermistor/ntc-thermistors-steinhart-and-hart-equation
- http://www.thinksrs.com/downloads/PDFs/ApplicationNotes/LDC%20Note%204%20NTC%20Calculator.pdf
- https://www.mundodaeletrica.com.br/sensor-de-temperatura-ntc-ptc/

Exemplo 7 - Módulo Relé 2 Canais

O Módulo Relé nada mais é do que um ou mais relés acionados por sinais digitais de 5V. Esse componente é indicado para acionar cargas que utilizam correntes contínuas maiores do que a suportada pelo Arduino, ou então uma carga de um circuito de corrente alternada (rede elétrica).

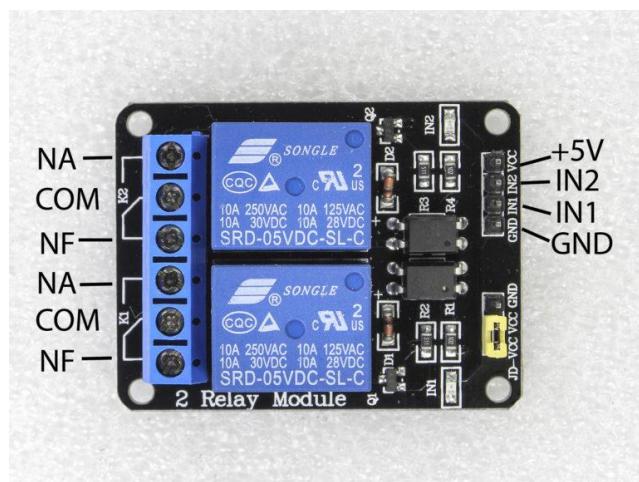
O módulo de relé pode ser usado para vários tipos de aplicações:

- ativação de eletroímãs,
- ativação de motores CC,
- ativação de motores CA,
- ligar/desligar lâmpadas.

No mercado você pode encontrar módulos de 1, 2, 4 e 8 canais. Cada canal corresponde a um relé montado no módulo. Ou seja, um módulo relé de 5V e 8 canais, é um conjunto de 8 relés acionados por sinais separados de 5V. Um módulo relé de 12V e 4 canais, nada mais do que um conjunto de 4 relés, cada qual acionado por sinais de 12V.

O módulo com 2 Relés tem um circuito de controle através de acopladores óticos. Isso é usado para isolar as portas do microcontrolador, do circuito de acionamento dos relés.

As portas de controle dos relés são os pinos IN1 e IN2. Por exemplo, para acionar o relé K1, a porta do microcontrolador conectada no pino IN1 deverá estar no nível baixo (LOW). Para controlar o relé K2, use o pino IN2.

O módulo funciona exatamente da mesma forma que uma chave ou interruptor. Esse é o diagrama de um dos relés SONGLE SRD-05VDC-SL-C:

- NA - Contato Normalmente Aberto
- C - Terminal Comum
- NF - Contato Normalmente Fechado

Existe um Led em série com cada entrada do acoplador ótico. Portanto quando a porta é ativada, o Led acende. A alimentação do módulo tem que ser 5 V (pino +5V). Não use uma tensão maior ou menor do que 5 V, pois poderá queimar o relé (tensão maior) ou então o relé não poderá ser acionado (tensão menor). Se quiser usar relés com 12 V, procure um outro módulo de relé. Não se esqueça de conectar o pino terra (GND) do módulo com o GND do Arduino. Mantenha o jumper entre os pinos JD-VCC e VCC. Dessa forma, a alimentação dos relés será de 5 V através do pino +5V. A capacidade corrente

em cada contato é de 10 A. Não ultrapasse esse valor, para não danificar o seu relé. O consumo de corrente da bobina de cada relé é de aproximadamente 70 mA, quando energizada.

Na imagem abaixo temos o Módulo Relé 2 canais com a identificação das conexões:

Módulo relé de 2 canais

Os pinos dos contatos dos Relés são NA (normalmente aberto), COM (comum) e NF (normalmente fechado.). Isto é, enquanto o relé estiver desativado, a ponta comum estará conectada no contato NF. Quando o relé for acionado, a ponta comum será conectada no contato NA.

Cada pino **INX** serve para acionar um dos relés. **VCC** e **GND** são os pinos de alimentação do Módulo do relé: VCC deve ser conectado no +5V e o GND no pino terra, ambos da fonte. **O pino terra (GND) do Módulo Relé deve estar conectado também no Terra (GND) do Arduino!**

Ponte H com Arduino

Ponte H é um tipo de circuito eletrônico muito usado para controlar um motor elétrico CC (corrente contínua). Através das portas de controle, pode-se ligar e desligar a energia do motor. E mais, alterar o sentido de rotação do motor.

Atualmente existem inúmeros módulos de Ponte H que usam chips dedicados. Mas um inconveniente é que para circuitos com maior corrente, os módulos são bem mais caros. Por isso criei esse tutorial para montagem de uma Ponte H com relés. A ponte H desse tutorial pode controlar motores CC com escova (Brushed) com corrente de até 10 amperes.

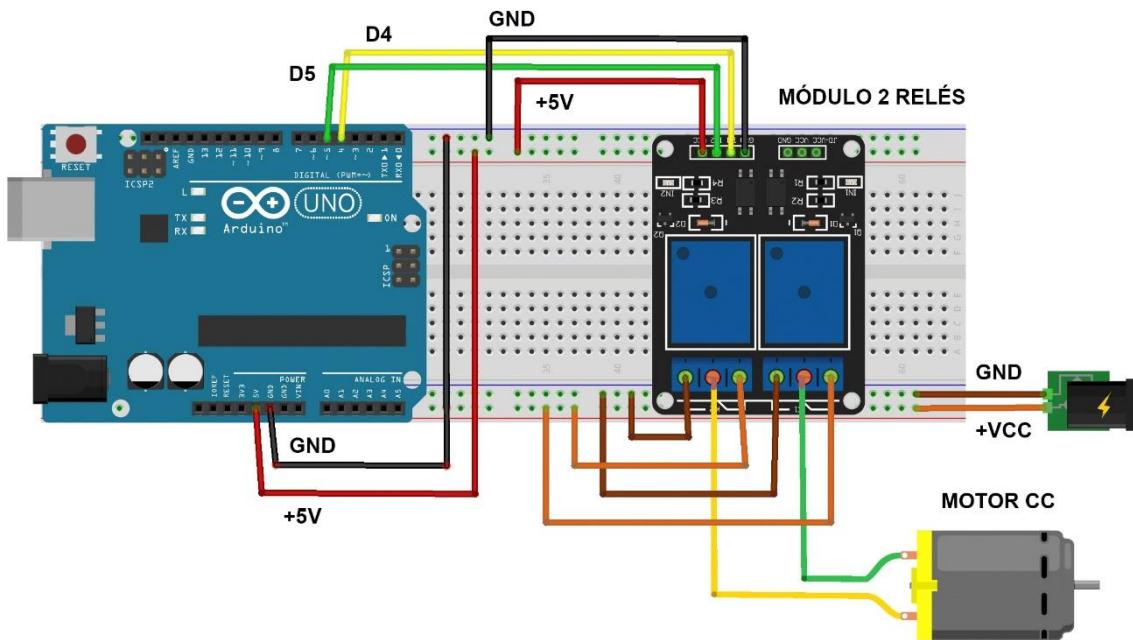
Uma limitação da Ponte H com relé, é que não suporta o controle de velocidade (PWM) do motor. Não é possível ficar chaveando um relé com uma frequência de pulsos. Isso danificaria os relés!

Para entender como uma ponte H funciona, recomendo a leitura do meu tutorial no Blog da Eletrogate:

http://blog.eletrogate.com/arduino-ponte-h-com-rele/

Lista de Materiais:

Para este exemplo você vai precisar:

- Protoboard,
- Arduino UNO,
- Jumpers de ligação,
- Módulo relé 2 canais,
- Motor CC,
- Fonte de alimentação.

O circuito abaixo é uma montagem de uma Ponte H básica com o Módulo 2 Relés. Veja que nessa montagem estou alimentando o motor com uma fonte externa CC. Verifique as especificações do seu motor, para saber qual é a tensão de alimentação e a corrente necessárias para o mesmo. Importante: a corrente de operação da fonte de alimentação precisa suportar a corrente do motor! Os relés suportam no máximo 10 A.

Para efeito didático, usaremos o Motor CC do Kit. Ele poderá ser alimentado com o +5V do Arduino, pois o consumo de corrente dele é bem baixo – aproximadamente 10 mA. Conecte o GND do conector ao GND do Arduino e o +VCC do conector ao +5V do Arduino. Não use motores CC com correntes maiores do que 50 mA, para não queimar o regulador 5V do Arduino.

Diagrama do circuito:

Esse é o programa de controle da Ponte H. O Arduino aciona um relé de cada vez, durante 3 segundos. Portanto o motor gira num sentido e depois gira no sentido inverso, sucessivamente. Essa montagem poderá ser aplicada em vários outros tipos de projetos, que necessitam o controle de acionamento de um motor CC. Sugiro que implemente botões para acionamento dos motores e chaves de limite de movimento (backstop) para desligar o motor, quando a parte móvel chegar no destino.

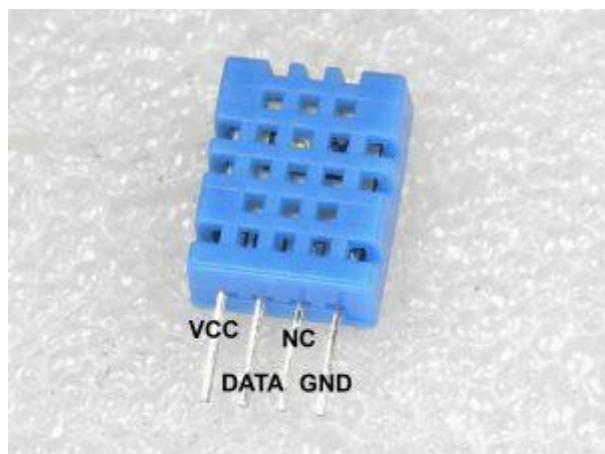
Código Arduino – Ponte H com módulo relé 2 canais

```
// Exemplo 7 - Ponte H com modulo 2 relés
// Apostila Eletrogate - KIT MAKER
// Gustavo Murta


int IN1 = 4;                          // pino IN1 conectado porta D4 Arduino
int IN2 = 5;                          // pino IN2 conectado porta D5 Arduino

void setup()
{
  Serial.begin(9600);                 // console serial 9600 Bps
  pinMode(IN1, OUTPUT);               // definições das portas como portas de saidas
  pinMode(IN2, OUTPUT);
  digitalWrite(IN1, HIGH);            // desativa porta IN1
  digitalWrite(IN2, HIGH);            // desativa porta IN2
}

void loop()
{
  digitalWrite(IN2, HIGH);            // desativa porta IN2
  digitalWrite(IN1, LOW);             // acionamento relé K1
  Serial.println("Rele K1 acionado"); // print mensagem
  delay (3000);                       // atraso de 3 segundos
  digitalWrite(IN1, HIGH);            // desativa porta IN1
  digitalWrite(IN2, LOW);             // acionamento relé K2
  Serial.println("Rele K2 acionado"); // print mensagem
  delay (3000);                       // atraso de 3 segundos
}
```

Referências:

- http://blog.eletrogate.com/modulo-rele-para-automacao-residencial-com-arduino/
- http://blog.eletrogate.com/arduino-ponte-h-com-rele/
- https://www.allaboutcircuits.com/projects/use-relays-to-control-high-voltage-circuitswwith-an-arduino/
- https://www.arduino.cc/en/Tutorial/Debounce

Exemplo 8 - Sensor De Umidade e Temperatura DHT11

O sensor DHT11 é um dispositivo de baixo custo usado para medição de umidade e temperatura do ar. O sensor de umidade é capacitivo e o sensor de temperatura é um termistor NTC, isto é um resistor sensível às variações de temperatura. Dentro do sensor existe um microcontrolador que faz as medições e transmite as medições no formato digital através de um pino de saída. Segundo o fabricante, a transmissão digital pode ser realizada através de um cabo de até 20 metros.

OBS = longa exposição do sensor ao sol, poderá afetar a performance do mesmo.

Especificações do sensor DHT11 (esses valores podem variar dependendo do fabricante):

- Faixa de umidade relativa = de 20 a 80 %
- Precisão na umidade = ± 5 % RH
- Resolução de umidade = 5 % RH
- Faixa de temperatura = 0 a 50 °C
- Precisão na temperatura = ± 2 % °C
- Tempo de resposta = 2 segundos
- Alimentação = de 3,5 V a 5 V
- Consumo máximo de corrente = 2,5 mA

Folha de especificações (Datasheet) do Sensor DHT11:

http://blog.eletrogate.com/wp-content/uploads/2018/12/DHT11-sumrom.pdf

Sensor DHT11 - pinagem

- VCC = 3,5 a 5V
- DATA = comunicação de dados

- NC = sem conexão
- GND = terra

Interface com o Sensor DHT11

Após a alimentação de tensão do Sensor DHT11, aguarde cinco segundos para o circuito se estabilizar. Um capacitor de 100 nF é recomendado entre o pino VCC e o GND, se a fiação for longa. No pino de saída, deve ser usado também um resistor de pullup -10 K ohms. Resistor Pullup é o resistor que fica conectado entre o pino de saída e o 5V.

A comunicação dos dados no barramento serial (único pino) ocorre nos dois sentidos, isto é, do sensor para o Arduino e vice-versa.

O protocolo de comunicação pode ser dividido em três partes:

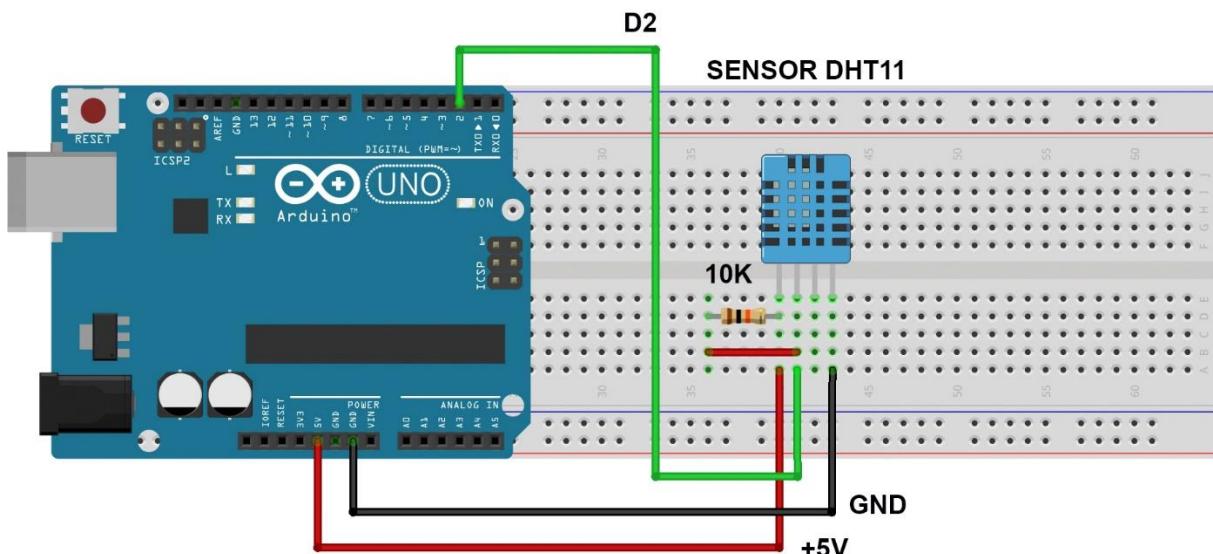
- Requisição: para o sensor enviar os dados, ele deverá ser requisitado,

- Resposta: o sensor envia uma resposta depois de requisitado,

- Leitura de dados: após a resposta do sensor, os dados são enviados em 5 segmentos de 8 bits (40 bits).

Se quiser saber mais sobre o sensor e o protocolo de comunicação, sugiro a leitura do documento abaixo:

Módulo sensor DHT11:
http://blog.eletrogate.com/wp-content/uploads/2018/12/DHT11-DFRobot.pdf

Sensor DHT11 com Arduino:

A montagem do Sensor DHT11 com Arduino é muito simples.

Lista de materiais:

- Arduino UNO,
- Protoboard,
- Sensor DHT11,
- Resistor de 10K,
- Jumpers para ligação no protoboard.

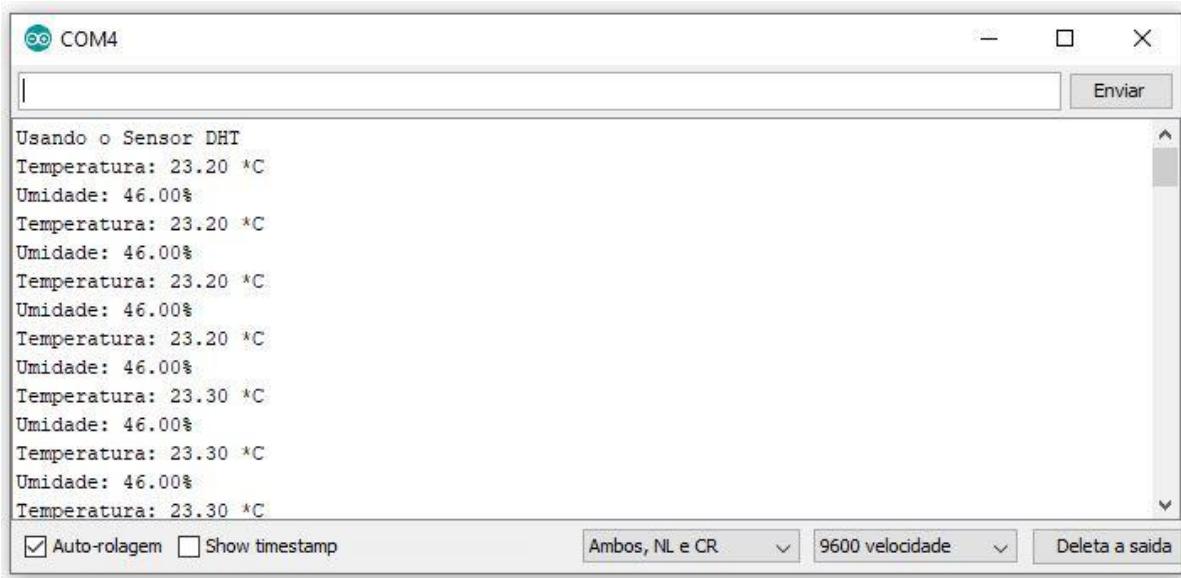
Diagrama do Circuito:

Instalando a Biblioteca DHT Sensor:

Para o uso do Sensor DHT11, a biblioteca que será usada é a da Adafruit. Ela segue os padrões da IDE Arduino e é bem simples a utilização.

Para instalar a Biblioteca **DHT Sensor Library** na Arduino IDE, clique em:

Sketch > Incluir Biblioteca > Gerenciar Bibliotecas

Após abrir a janela do Gerenciador de Biblioteca, refine a busca digitando DHT Sensor Library. Na biblioteca da Adafruit, clique em **More Info** e depois em **Instalar**. Após alguns segundos, ela será automaticamente instalada. Lembre-se que o seu computador precisa estar conectado na internet, para poder baixar a biblioteca. Após a instalação da Biblioteca, é necessário que feche e abra novamente o programa da Arduino IDE.

Sketch Arduino – Sensor DHT

O programa também é bem simples. Fiz uma tradução do exemplo da Biblioteca da Adafruit, para facilitar o entendimento. O programa faz com que o Arduino se comunique com o Sensor DHT, para realizar as leituras de umidade e temperatura.

OBS: Veja o Sketch na próxima página.

Janela da Console com as medições:

Referências:

- http://blog.eletrogate.com/guia-basico-dos-sensores-de-umidade-e-temperatura-dht11-dht22/
- https://learn.adafruit.com/dht/overview
- https://create.arduino.cc/projecthub/search?q=dht11
- https://playground.arduino.cc/Main/DHT11Lib/

Código Arduino – Sensor DHT11 – Umidade e Temperatura:

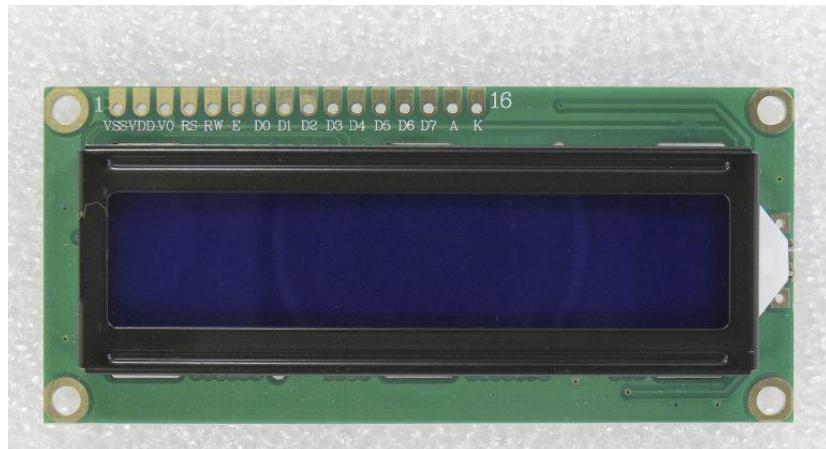
```
// Exemplo 8 - Arduino - Sensor DHT11 - Umidade e Temperatura
// Apostila Eletrogate - KIT MAKER
// Gustavo Murta

#include <Adafruit_Sensor.h>                    // Biblioteca DHT Sensor Adafruit
#include <DHT.h>
#include <DHT_U.h>

#define DHTTYPE    DHT11                        // Sensor DHT11

#define DHTPIN 2                                // Pino do Arduino no Sensor (Data)
DHT_Unified dht(DHTPIN, DHTTYPE);               // configurando o Sensor DHT - pino e tipo
uint32_t delayMS = 500;                         // variável para atraso no tempo

void setup()
{
  Serial.begin(9600);                           // monitor serial 9600 bps
  dht.begin();                                  // inicializa a função
  Serial.println("Usando o Sensor DHT");
  sensor_t sensor;
}

void loop()
{
  delay(delayMS);                               // atraso entre as medições
  sensors_event_t event;                        // inicializa o evento da Temperatura
  dht.temperature().getEvent(&event);           // faz a leitura da Temperatura
  if (isnan(event.temperature))                 // se algum erro na leitura
  {
    Serial.println("Erro na leitura da Temperatura!");
  }
  else                                          // senão
  {
    Serial.print("Temperatura: ");              // imprime a Temperatura
    Serial.print(event.temperature);
    Serial.println(" *C");
  }
  dht.humidity().getEvent(&event);              // faz a leitura de umidade
  if (isnan(event.relative_humidity))           // se algum erro na leitura
  {
    Serial.println("Erro na leitura da Umidade!");
  }
  else                                          // senão
  {
    Serial.print("Umidade: ");                  // imprime a Umidade
    Serial.print(event.relative_humidity);
    Serial.println("%");
  }
}
```

Exemplo 9 - Display LCD 16x2

O display 16×2 é um dispositivo que possui interface para comunicação e controle padronizada. Esses displays são fáceis de serem controlados e graças à essa padronização é possível trocar um display 16×2 de um fabricante por outro diferente sem maiores problemas.

A tecnologia utilizada nos displays tradicionais é LCD (Liquid Crystal Display). Mais recentemente, modelos mais modernos com tecnologia O-Led, os LEDS orgânicos, já estão sendo encontrados também.

A grande vantagem e razão pela qual até hoje é componente popular, é o seu preço e a facilidade de implementação em projetos eletrônicos. Antes dos displays LCD, a interface gráfica usada para mostrar caracteres alfa numéricos era os displays a LED de x segmentos (os mais comuns eram de 7, 14 ou 16 segmentos).

Esses componentes mais antigos utilizavam vários leds (7,14 ou 16) dispostos em segmentos, de forma que dependendo de qual conjunto de leds fosse aceso, um determinado caractere seria mostrado.

LCD 16x2. Créditos: Eletrogate

Na hora de comprar um display LCD, você vai encontrar diversas opções, além do mais tradicional 16×2. A especificação é dada em relação ao número de caracteres por linha e o número de linhas. Assim, 16×2 significa que o display pode exibir 16 caracteres em cada linha, e possui 2 linhas. Alguns valores típicos para essas especificações são:

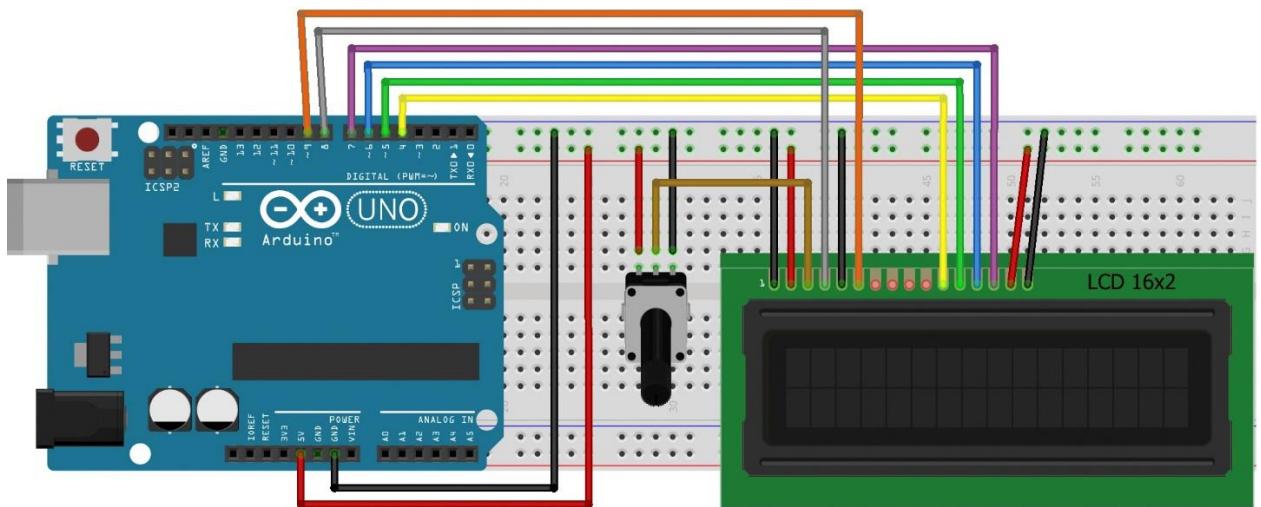
- Quantidade de caracteres padrão de mercado: 8, 12, 16, 20, 24 e 40;
- Quantidade de linhas mais comuns: 1, 2 e 4;
- Cores de fundo (backlight): Azul ou verde;

O display que vamos trabalhar é de 16 colunas e 2 linhas (16×2) e com backlight azul e caracteres brancos. A interface com Arduino é feita por meio de 16 pinos. Em geral apenas 12 são utilizados de fato. Os pinos do LCD são:

LCD 16x2 pinagem – foto Gustavo Murta

- Pino de **register select (RS):** Controle em que parte da memória (do LCD) o usuário está escrevendo os dados. São duas opções, você pode escrever um dado para ser mostrado no display, ou uma instrução no MCU do display;
- Pino de **Read/Write (R/W**): Seleciona se está em modo de leitura ou de escrita;
- Pino de **Enable**: Habilita a escrita de dados nos registradores;
- 8 pinos de dados **(D0 -D7).** Os estados de cada pino desses corresponde aos bits que você está escrevendo em um registrador do MCU (do display), ou os valores que você lê de um registrador quando está no modo de leitura.
- Pinos de alimentação **VCC e GND** do LCD e do LED Backlight (A-anodo/K-catodo);
- Pino **V0** de ajuste de contraste;

Vamos conferir em nosso exemplo como usar o LCD 16x2. Na figura abaixo mostramos um outro modelo, um display gráfico de 128 colunas e 64 linhas, também muito comercializado e que você pode obter separadamente do Kit para implementar mensagens gráficas mais avançadas.

Display gráfico 128x64

Lista de materiais:

A lista de materiais é a seguinte:

- 1 Arduino UNO;
- 1 Protoboard;
- Jumpers para ligação no protoboard;
- Display LCD 16x2;
- Potenciômetro 10K;

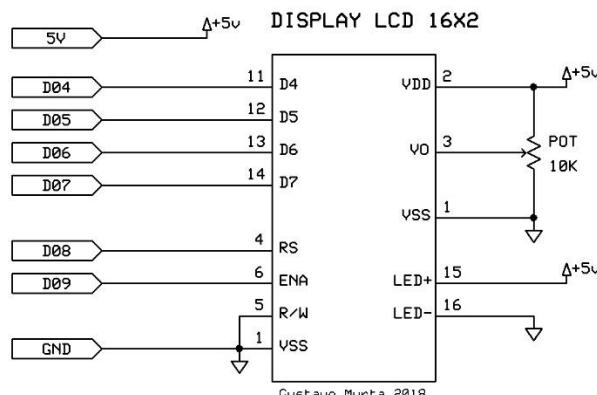



Diagrama do circuito:

No diagrama acima, o potenciômetro é usado para controlar o contraste do LCD, e deve ser ajustado sempre que você ligar o display para poder visualizar bem cada carácter. As demais ligações são padrão. No código da próxima seção, os pinos do Arduino são configurados de acordo com a ligação acima.

Veja o diagrama eletrônico da montagem acima:

Código Arduino – Display LCD 16x2:

```
// Exemplo 9 - Display LCD 16x2
// Apostila Eletrogate - KIT MAKER

#include <LiquidCrystal.h>                // biblioteca Liquid Crystal

LiquidCrystal lcd(8, 9, 4, 5, 6, 7);      // pinos do LCD - RS E D4 D5 D6 D7
int contador = 0;                         // variável contador

void setup()
{
  lcd.begin(16, 2);                       // inicializa LCD 16x2
  delay(500);                             // atraso de 0,5 segundos
}

void loop()
{
  lcd.clear();                            // limpa tela do LCD
  lcd.setCursor(0, 0);                    // selecionando coluna 0 e linha 0
  lcd.print("Exemplo LCD !");             // mostra no LCD
  lcd.setCursor(1, 1);                    // selecionando coluna 1 e linha 1
  lcd.print(contador);                    // mostra no LCD a contagem
  contador++;                             // incrementa contador
  if (contador == 60)                     // se contador = 60
    contador = 0;                         // zera o contador
  delay(1000);                            // atraso de 1 segundo
}
```

No código, é mostrado na primeira linha (0) do display, a mensagem “Exemplo LCD!”. Na segunda linha (1) será mostrado um contador que é incrementado de 1 em 1 segundo. Para usar o LCD é preciso incluir a biblioteca **LiquidCrystal.h**, que é nativa da IDE arduino.

Essa biblioteca possui as seguintes funções:

- lcd.clear() - Limpa a tela do display
- lcd.SetCursor(0,0) - Posiciona o cursor a partir de onde os caracteres serão escritos
- lcd.print() - Imprime caracteres no display

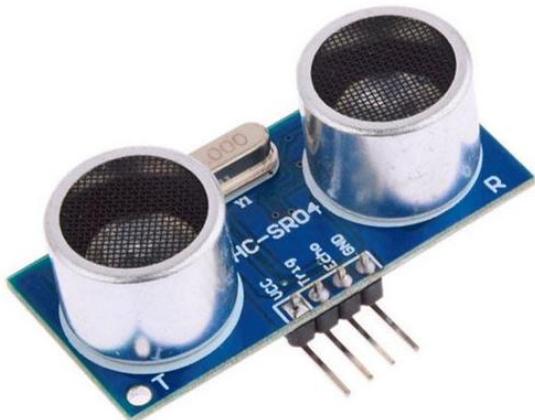
No exemplo são usadas essas três funções, mas a biblioteca dispõe de várias outras que você pode conferir na página oficial (vejas as referências abaixo).

Referências:

- http://blog.eletrogate.com/guia-completo-do-display-lcd-arduino/
- http://blog.eletrogate.com/display-lcd-modulo-rf-para-deteccao-de-presenca/
- https://www.arduino.cc/en/Reference/LiquidCrystal

Exemplo 10 - Sensor de Distância Ultrassônico HC-SR04

O princípio de funcionamento do Sensor HC-SR04 consiste na emissão de sinais ultrassônicos e na leitura do sinal de retorno (reflexo/eco) desse mesmo sinal. A distância entre o sensor e o objeto que refletiu o sinal é calculada com base no tempo entre o envio e leitura de retorno.

"Sinais Ultrassônicos são ondas mecânicas com frequência acima de 40 KHz"

Como ouvido humano só consegue identificar ondas mecânicas até a frequência de 20KHz, os sinais emitidos pelo sensor Ultrassônico não podem ser escutados por nós.

O sensor HC-SR04 é composto de três partes principais:

- Transmissor Ultrassônico – Emite as ondas ultrassônicas que serão refletidas pelos obstáculos;
- Um receptor – Identifica o eco do sinal emitido pelo transmissor;
- Circuito de controle – Controla o conjunto transmissor/receptor, calcula o tempo entre a emissão e recepção do sinal;

O sensor é ilustrado na imagem abaixo. Repare como os invólucros do transmissor e receptor são bem proeminentes.

Sensor HC-SR04

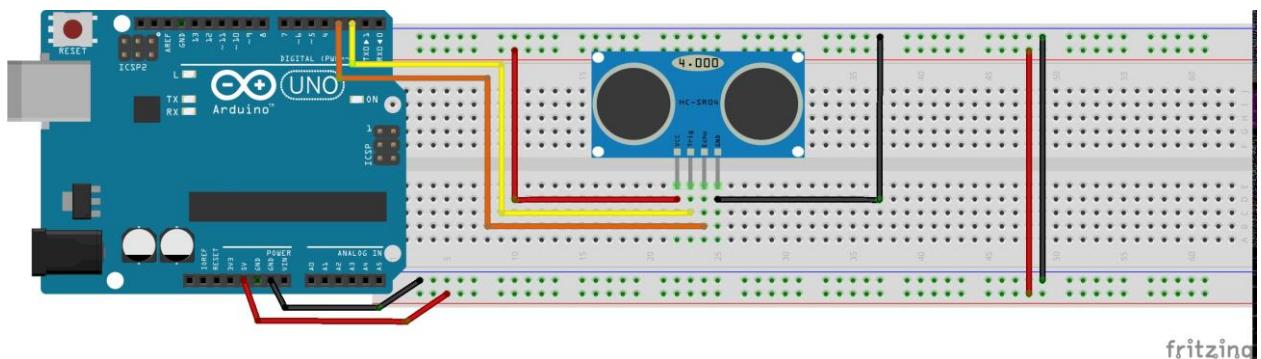
O funcionamento consiste em três etapas:

1. O circuito externo (Arduino) envia um pulso de trigger de pelo menos 10us em nível alto;
2. Ao receber o sinal de trigger, o sensor envia 8 pulsos de 40khz e detecta se há algum sinal de retorno ou não;
3. Se algum sinal de retorno for identificado pelo receptor, o sensor gera um sinal de nível alto no pino de saída cujo tempo de duração é igual ao tempo calculado entre o envio e o retorno do sinal ultrassônico;

Por meio desse pulso de saída cujo tempo é igual a duração entre emissão e recepção, nós calculamos a distância entre o sensor e o objeto/obstáculo, por meio da seguinte equação:

$$Distancia = \frac{(Tempodeduraçãodosinaldesaídaxvelocidadedosom)}{2}$$

Em que o Tempo de duração do sinal de saída é o tempo no qual o sinal de output permanece em nível alto e a velocidade do som = 340 m/s;

Repare que as unidades devem estar coerentes para o resultado da conta ser correto. Assim, se a velocidade do som é dada em metros/segundo, o tempo de duração do sinal de saída deve estar em segundos para que possamos encontrar a distância em metros.

Lista de Materiais:

Para este exemplo você vai usar os seguintes componentes:

- Arduino UNO,
- Protoboard,
- Jumpers de ligação,
- Sensor ultrassônico HCSR-04.

Diagrama do circuito:

A montagem é bem direta, basta interligar o sensor com o Arduino. Não há necessidade de conversão de níveis lógicos ou ligação de componentes externos.

Código Arduino – Sensor Ultrassônico HC-SR04:

As variáveis declaradas são para determinar os pinos de trigger (pino 2) e de leitura do sensor (pino 3). Temos três variáveis do tipo float utilizadas para medir o tempo do pulso no output do sensor e calcular a distância. Na função void setup(), inicializamos o pino 2 como saída e o 3 com entrada. Além disso, configuramos a comunicação serial para 9600 de baud rate.

Na função void loop(), onde o programa será executado continuamente, executa-se três passos básicos:

1. Enviar pulso de 10us para o pino de trigger do sensor. Isto é feito com a função DisparaPulsoUltrassonico();

2. Ler o pino de output do sensor e medir o tempo de duração do pulso utilizando a função PulseIn. Esta função retorna o tempo de duração do pulso em microsegundos para que o mesmo seja armazenado em uma variável;

3. Por fim, chamamos a função CalculaDistancia (tempo) passando como parâmetro o tempo lido com a função PulseIn. Esta função calcula a distância entre o sensor e o obstáculo. Nós chamamos esta função de dentro da função Serial.println(), de forma que o resultado já é impresso diretamente no terminal serial;

```
// Exemplo 10 - Sensor de Distância Ultrassônico HC-SR04
// Apostila Eletrogate - KIT Maker

int PinTrigger = 2;                              // pino usado para disparar os pulsos do sensor
int PinEcho = 3;                                 // pino usado para ler a saida do sensor
float TempoEcho = 0;                             // variável tempo do eco
const float velocidadeSom_mps = 340;             // em metros por segundo
const float velocidadeSom_mpus = 0.000340;       // em metros por microsegundo

void setup()
{
  pinMode(PinTrigger, OUTPUT);                   // configura pino Trigger como saída
  digitalWrite(PinTrigger, LOW);                 // pino trigger - nível baixo
  pinMode(PinEcho, INPUT);                       // configura pino ECHO como entrada
  Serial.begin(9600);                            // inicializa monitor serial 9600 Bps
  delay(100);                                    // atraso de 100 milisegundos
}

void loop()
{
  DisparaPulsoUltrassonico();                         // dispara pulso ultrassonico
  TempoEcho = pulseIn(PinEcho, HIGH);                 // mede duração do pulso HIGH de eco em micro seg
  Serial.print("Distancia em metros: ");              // mostra no monitor serial

  Serial.println(CalculaDistancia(TempoEcho));        // mostra o calculo de distancia em metros
  Serial.print("Distancia em centimentros: ");        // mostra no monitor serial
  Serial.println(CalculaDistancia(TempoEcho) * 100); // mostra o calculo de distancia em cm
  delay(2000);                                        // atraso de 2 segundos
}
//CONTINUA NA PRÓXIMA PÁGINA
```

```
void DisparaPulsoUltrassonico()
{
  digitalWrite(PinTrigger, HIGH);                   // pulso alto de Trigger
  delayMicroseconds(10);                            // atraso de 10 milisegundos
  digitalWrite(PinTrigger, LOW);                    // pulso baixo de Trigger
}

float CalculaDistancia(float tempo_us)
{
  return ((tempo_us * velocidadeSom_mpus) / 2 );    // calcula distancia em metros
}
```

A função **DisparaPulsoUltrassonico()** apenas ativa o pino 2 em nível alto, espera 10 microsegundos e volta a setar o pino para nível baixo. Lembre-se que 10 us é o tempo mínimo que o pulso deve perdurar para disparar o sensor HC-SR04.

A função **CalculaDistancia()** recebe como parâmetro o tempo do pulso no pino Echo do sensor. Com esse tempo nós usamos a equação, para calcular a distância entre o sensor e o obstáculo.

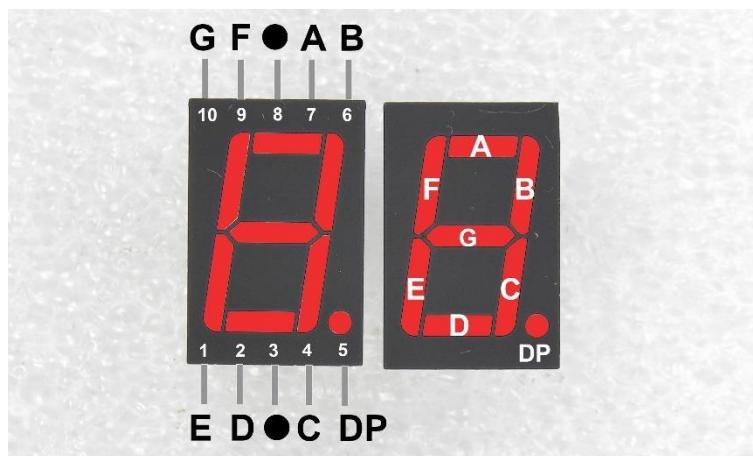
Referências:

- https://create.arduino.cc/projecthub/dusnoki/arduino-ultrasonic-sensor-hc-sr04-full-build-and-code-73614d
- http://blog.eletrogate.com/sensor-ultrassonico-hc-sr04-com-arduino/

Exemplo 11 – Display de 7 segmentos + Teclado de membrana

Para alguns tipos de aplicações dos Microcontroladores é necessária uma interface de entrada de dados, que permita a comunicação entre o homem e a máquina. E a interface mais usada é o teclado. Através de um teclado, o usuário da “máquina” pode selecionar e alterar funções ou operações.

Os teclados de membrana são muito usados ainda, pois eles têm baixo custo, boa durabilidade e boa confiabilidade nas operações. Todos os celulares usam os teclados de membrana. Os smartphones mais antigos usavam também, mas atualmente os teclados são capacitivos e montados sobre a tela. Os teclados de laptops e desktops usam o mesmo princípio de teclado matricial, mas no lugar das membranas são usadas teclas.

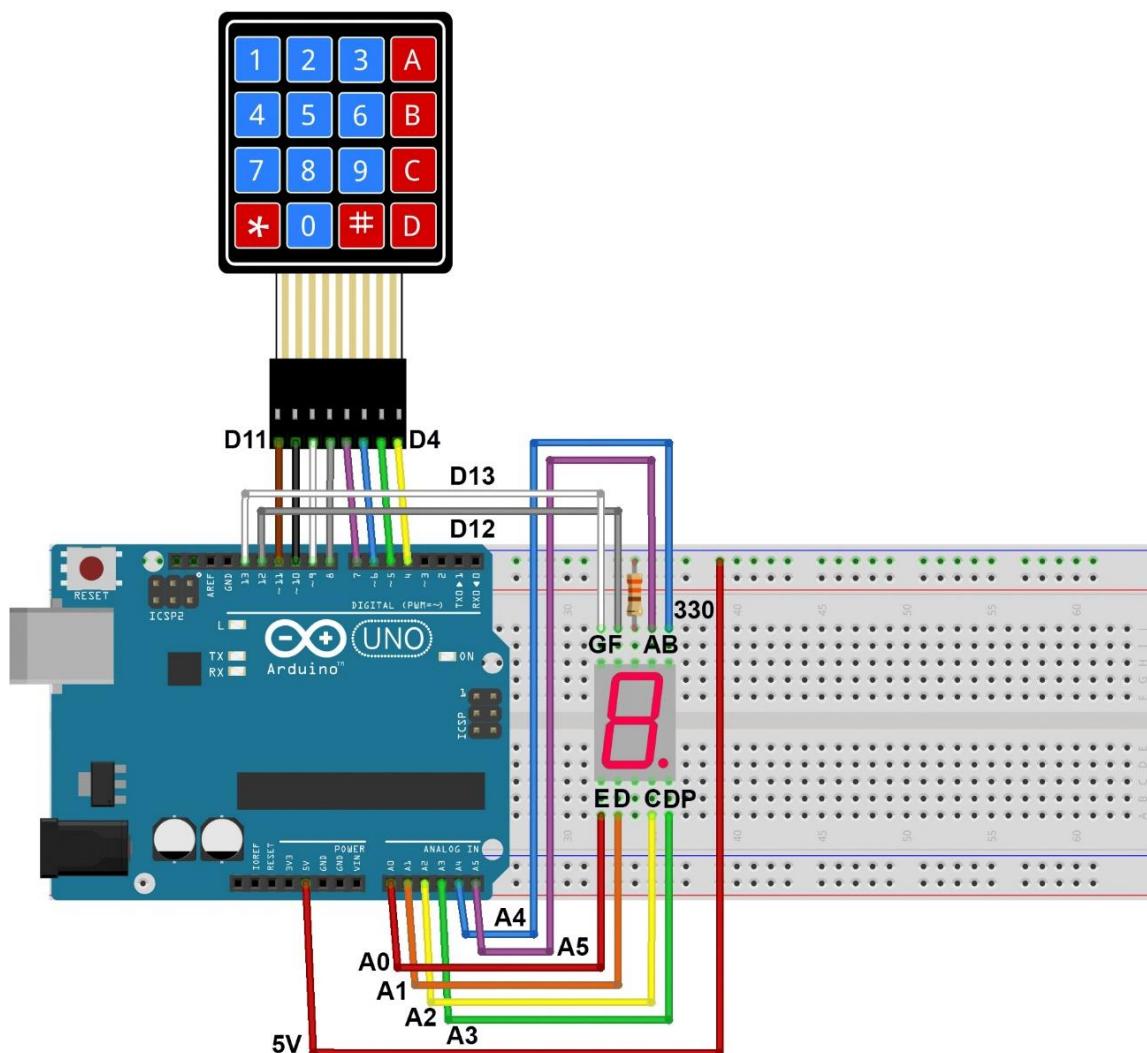
Esse teclado de membrana é montado como uma matriz de teclas. Por isso também é chamado de matricial. Mas o nome mais usado em inglês é Keypad, devido ao pequeno tamanho. Veja a identificação dos pinos do conector (L = linhas e C = colunas). Essa foto foi editada para diminuir o tamanho do cabo.

Os displays de 7 segmentos foram desenvolvidos a partir da evolução dos Leds. Como já sabe, o Led é um diodo de material semicondutor que quando a corrente elétrica é passada por ele, uma luz é emitida. Devido à construção desse semicondutor, é necessária uma limitação de corrente, pois senão ele queima. Por isso, para a ligação de um Led, um resistor em série deve ser conectado.

Perceba que cada segmento do Display é formado por um led. Ao acionar os segmentos, pode-se formar números e letras (com algumas limitações). É esse o princípio simples de funcionamento de um Display. Os segmentos são identificados com letras de A até G e mais DP para o ponto decimal.

Existem dois tipos de displays, com o catodo comum e com o anodo comum. Como o nome já diz, no primeiro tipo, os catodos de todos os segmentos estão conectados em um único pino negativo. E no segundo tipo, os anodos estão conectados também em um só pino, mas positivo. Portanto as polarizações dos segmentos são inversas uma da outra.

Esse é o display de 7 segmentos – anodo comum (F5161BH) utilizado nessa montagem. Os pinos com bolinhas pretas são os pinos comuns – anodos. Use somente um, já que os dois são interligados internamente. Cada segmento é identificado com uma letra – A até G. E o ponto decimal, não usado nessa montagem, é o DP.

Montagem Arduino com Display 7 segmentos e teclado de membrana:

Logo abaixo temos o diagrama da montagem. O keypad foi conectado nos pinos D4 até D11. E o display de 7 segmentos foi conectado nas portas analógicas A0 a A5 e nos pinos D12 e D13. Como nessa montagem uma grande quantidade de portas é necessária, tive que usar as portas analógicas também. Não sei se já sabia, mas essas portas que normalmente são utilizadas para medições analógicas, podem ser usadas como portas digitais de entrada e saída. Foi isso que eu fiz, usei essas portas como

saída, controlando o display. Como o display é do tipo anodo comum, esse pino comum foi conectado ao 5V do Arduino, acrescentando um resistor de 330 ohms em série para limitar a corrente nos segmentos.

Lista de Materiais:

Para este exemplo você vai usar os seguintes componentes:

- Arduino UNO,
- Protoboard,
- Jumpers de ligação,
- Teclado de membrana,
- Resistor 330 ohms,
- Display 7 segmentos – anodo comum.

Diagrama do Display e do teclado:

Código Arduino - Display 7 segmentos e teclado de membrana:

O programa que eu desenvolvi tem o objetivo de identificar a tecla pressionada e mostrá-la no display. Como no display não existem os caracteres * e #, identifiquei-os como F e E respectivamente.

A biblioteca usada para controlar o Keypad tem o mesmo nome. Para instalar a nova Biblioteca na Arduino IDE, clique em:

Sketch > Incluir Biblioteca > Gerenciar Bibliotecas

Após abrir a janela do Gerenciador de Biblioteca, refine a busca digitando **keypad** . Na biblioteca **Keypad** (de Mark Stanley), clique em **More Info** e depois em **Instalar.** Após alguns segundos, ela será automaticamente instalada. Lembre-se que o seu computador precisa estar conectado na internet, para poder baixar a biblioteca. Após a instalação da Biblioteca, é necessário que feche e abra novamente o programa Arduino IDE.

A biblioteca **SevenSegment** usada para controlar o display deverá ser instalada de um modo diferente. Faça o download do arquivo **SevenSegment-master.zip**, do link abaixo:

http://blog.eletrogate.com/wp-content/uploads/2019/04/SevenSegment-master.zip

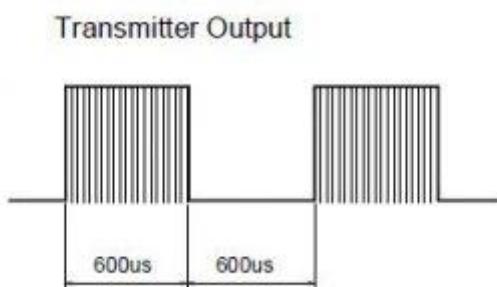
Clique em: **Sketch > Incluir Biblioteca > Adicionar biblioteca.zip**

Abra o arquivo no seu computador, para instalar automaticamente a biblioteca. Após a instalação da Biblioteca, é necessário que feche e abra novamente o programa Arduino IDE.

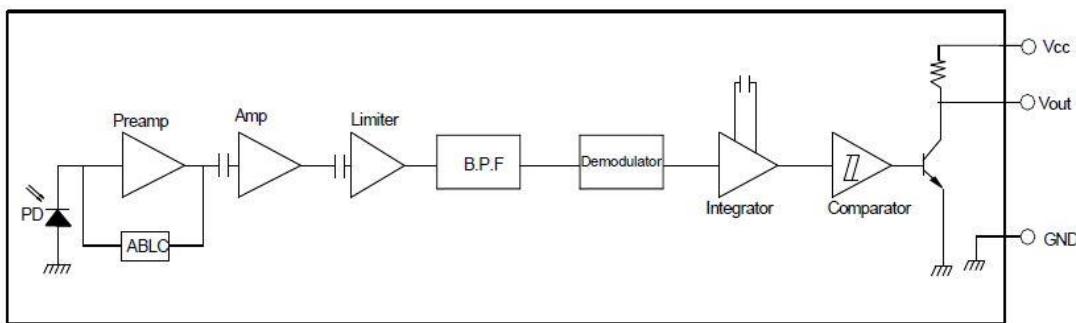
Código Arduino - KeypadDisplayDigital:

```
// Exemplo 11 – Display de 7 segmentos + Teclado de membrana
// Apostila Eletrogate - KIT MAKER
// Gustavo Murta


#include <Keypad.h>                        // biblioteca Keypad
#include <SegmentDisplay.h>                // biblioteca SegmentDisplay

SegmentDisplay segmentDisplay(A0, A1, A2, A3, A4, A5, 12, 13);
 // definição das portas para segmentos do display
 // A0=E A1=D A2=C A3=D A4=B A5=A D12=F D13=G

const byte ROWS = 4;                       // Keypad 4 linhas
const byte COLS = 4;                       // Keypad 4 colunas

char hexaKeys[ROWS][COLS] =                // definição dos caracteres do Keypad
{
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'F', '0', 'E', 'D'}                     // alterações *=F e #=E
};

byte rowPins[ROWS] = {11, 10, 9, 8};     // portas D11 a D8 => linhas do Keypad
byte colPins[COLS] = {7, 6, 5, 4};       // portas D7 a D4  => colunas do Keypad

Keypad customKeypad = Keypad( makeKeymap(hexaKeys), rowPins, colPins, ROWS,
COLS);                                   // inicializa o Keypad

void setup()
{
  Serial.begin(9600);                    // console serial 9600 Bps
 // definições das portas analógicas A0 a A5 como portas digitais de saidas
  pinMode(A0, OUTPUT
  pinMode(A1, OUTPUT);
  pinMode(A2, OUTPUT);
  pinMode(A3, OUTPUT);
  pinMode(A4, OUTPUT);
  pinMode(A5, OUTPUT);
}

void loop()
{
  char customKey = customKeypad.getKey();  // leitura da tecla pressionada
  int number = customKey - 48;             // converte o caracter em um número

  if ((number > 16) && (number < 23)) number = number - 7;
  // se numero for entre 17 e 22 subtraia 7 (numeros hexadecimais)

  if (customKey)                           // se tecla for pressionada
  {
    segmentDisplay.displayHex(number, false);
    // mostra o caracter no display sem o ponto decimal
    Serial.println(number);
   // imprime o numero na console serial (para teste)
  }
}
```

Referências :

- http://blog.eletrogate.com/arduino-keypad-4x4-e-display-digital/
- http://blog.eletrogate.com/guia-completo-dos-displays-de-7-segmentos-arduino/
- https://www.explainthatstuff.com/computerkeyboards.html
- https://www.nutsvolts.com/magazine/article/using-seven-segment-displays-part-1
- https://www.nutsvolts.com/magazine/article/using-seven-segment-displays-part-2

Exemplo 12 – Controle Remoto Ir + Receptor Universal Ir

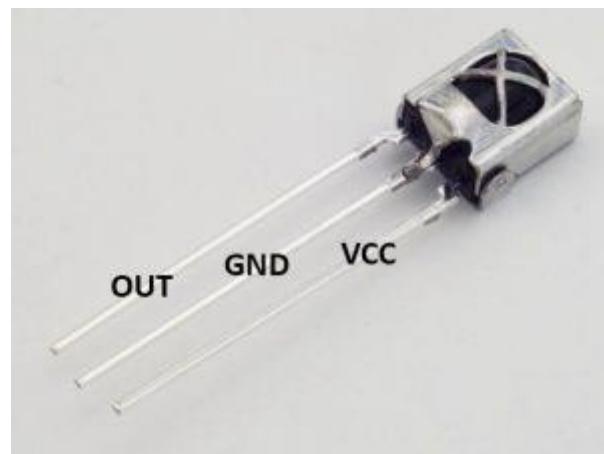
O controle Remoto IR (infrared, ou infravermelho em português) de aparelhos eletrônicos consiste em um pequeno dispositivo que contém um chip de microcontrolador, um ou mais LEDs emissores de infravermelho e um teclado acoplado. Quando o usuário pressiona uma das teclas do controle, uma sequência de pulsos de luz infravermelha é transmitida pelos LEDs. Esses pulsos formam um código que é único para cada tecla acionada.

O chip do Microcontrolador do Controle Remoto Transmissor, já vem programado com um protocolo definido pelo fabricante do aparelho.

Os pulsos de dados (código) são enviados do transmissor através de outros pulsos numa frequência bem maior, modulando o sinal a ser transmitido. Alguns trabalham com frequências diferentes de modulação, mas a mais comum atualmente é a frequência de 38 KHz. Na figura abaixo, pode-se perceber que quando o pulso de dado é 1, a frequência é transmitida e quando o pulso é 0, nenhuma frequência é transmitida. Esse tipo de modulação chama-se PCM (modulação codificada de pulsos). Com esse tipo de modulação, impede-se que a luz externa interfira na transmissão dos dados.

O circuito Receptor IR que fica internamente no aparelho a ser controlado, consiste basicamente de um descodificador dos pulsos recebidos do transmissor. Interpretando o código de cada tecla pressionada, o aparelho pode executar vários tipos de operações, como por exemplo, desligar, aumentar volume e mudar o canal no caso de uma Televisão.

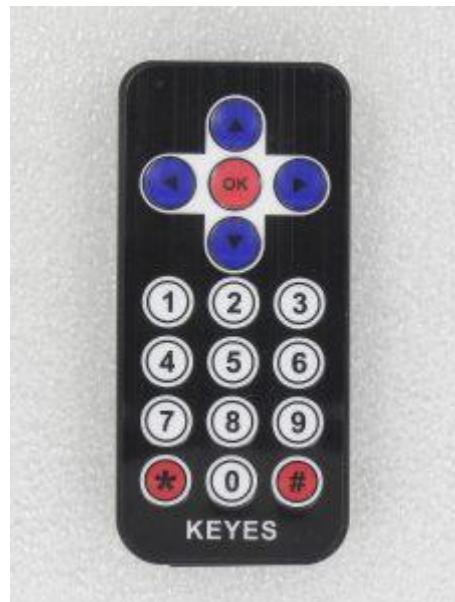
Hoje com a integração cada vez maior dos circuitos, existem receptores de controle remoto IR que já contém o foto-sensor e todos os outros circuitos necessários para demodular os pulsos de controle, como amplificadores de sinal, filtros, demodulador, integrador e comparador.

O diagrama de blocos abaixo contém os vários circuitos que estão integrados no pequeno receptor. Após a demodulação dos pulsos, esses são descodificados por um outro Microcontrolador, para que o aparelho possa entender que tipo de operação deverá ser realizada.

Receptor de Infravermelho AX-1838HS

Esse é foto-receptor IR usado nesse exemplo. Ele é amplamente utilizado em aparelhos para permitir o controle remoto. Pode ser encontrado em TVs, rádios, aparelhos de multimídia e até em aparelhos de ar condicionado.

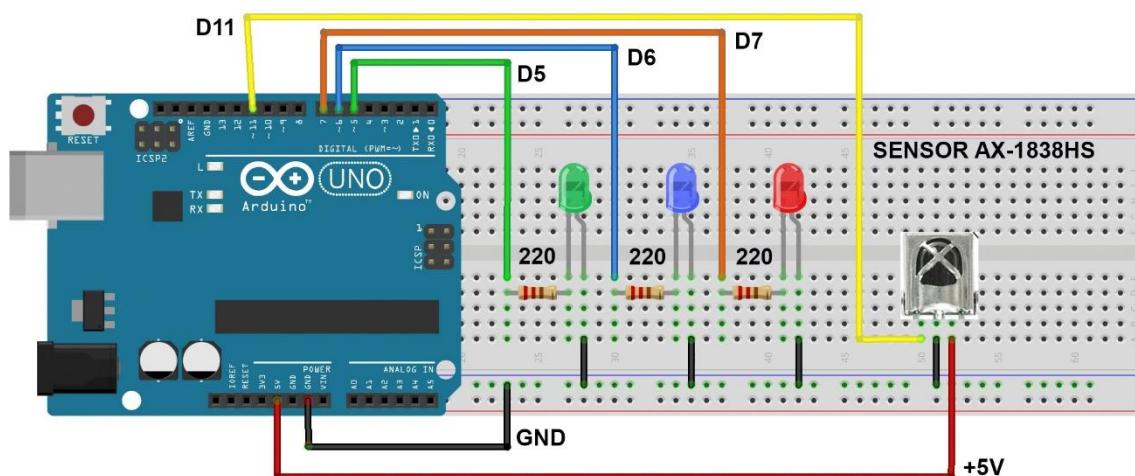
Ele pode trabalhar com tensões de 2,1V a 5,5V. O consumo de energia é bem baixo, com uma corrente de aproximadamente 1,5 mA. A frequência de modulação dos pulsos de dados é de 38 KHz. E o comprimento de onda da luz infravermelha percebida é de 940 nm. O ângulo de visão é de aproximadamente 90 graus. A vantagem do uso desse foto-receptor IR é que ele já tem todo o circuito integrado necessário para demodular os pulsos.

Folha de especificações (Datasheet) do AX-1838HS:

http://blog.eletrogate.com/wp-content/uploads/2018/12/IR-Receiver-AX-1838HS.pdf

Sensor AX-1838HS - pinagem

Controle Remoto Infravermelho:

O controle remoto IR (transmissor) é o da KEYES com 17 botões. Ele funciona com uma pilha botão CR2025. Antes de usá-lo, retire a proteção de plástico transparente que fica dentro do suporte da bateria. Para trocar a pilha, pressione a trava para o lado e puxe o suporte da pilha. Veja o diagrama na parte debaixo do controle remoto. Esse controle remoto usa o protocolo de codificação da NEC.

Controle Remoto IR com Arduino:

Usando um circuito de Controle Remoto IR com o Arduino, poderá implementar inúmeras aplicações bem interessantes como:

- Controle remoto de lâmpadas (liga/desliga ou até controle de brilho),
- Acionamento remoto de aparelhos,
- Controle remoto de alarmes,
- Controle remoto de câmera fotográfica.

A montagem do circuito é bem simples, interligando um sensor Receptor IR AX-1838Hs ao Arduino. A alimentação do sensor é fornecida pelo 5V do Arduino. E o pino de saída dos pulsos do Receptor é conectado ao pino D11 do Arduino. O Led verde está conectado no pino D05, o Led azul no pino D06 e o Led vermelho no pino D07 do Arduino. Todos Leds tem um resistor de 220 ohms em série com as ligações. O lado chanfrado do Led é o catodo, que deve ser conectado ao pino terra (GND).

Diagrama do Controle Remoto IR para Arduino:

Instalando a Biblioteca IRremote:

Para instalar a Biblioteca **IRremote**, clique em:

Sketch > Incluir Biblioteca > Gerenciar Bibliotecas

Após abrir a janela do Gerenciador de Biblioteca, refine a busca digitando **IRremote**. Na biblioteca **IRremote** (de shiirf), clique em **More Info** e depois em **Instalar.** Após alguns segundos, ela será automaticamente instalada. Lembre-se que o seu computador precisa estar conectado na internet, para poder baixar a biblioteca. Após a instalação da Biblioteca, é necessário que feche e abra novamente o programa Arduino IDE.

O sketch do exemplo é bem interessante, pois com o uso do controle remoto poderá ligar ou desligar os LEDs coloridos. Se quiser acionar um equipamento, poderá substituir os LEDs por um módulo de relé.

- Tecla 1 controla Led Verde / pino D05
- Tecla 2 controla Led Azul / pino D06
- Tecla 3 controla Led Vermelho / pino D07

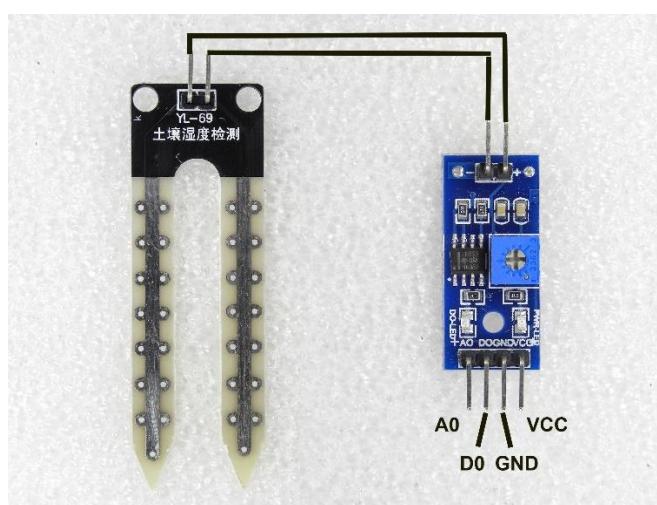
Veja o Código Arduino na próxima página.

Referências :

- http://blog.eletrogate.com/guia-completo-do-controle-remoto-ir-receptor-ir-para-arduino/
- https://learn.adafruit.com/using-an-infrared-library/overview
- http://labdegaragem.com/profiles/blogs/6223006:BlogPost:315534
- https://hetpro-store.com/TUTORIALES/control-ir-con-arduino/
- http://www.righto.com/search/label/ir

Código Controle Remoto IR - LEDs coloridos:

```
// Exemplo 12 – Controle Remoto IR - LEDs coloridos
// Apostila Eletrogate - KIT MAKER
// Gustavo Murta

#include <IRremote.h>                                    // Biblioteca IRemote
int RECV_PIN = 11;                                       // Arduino pino D11 conectado no
Receptor IR
IRrecv irrecv(RECV_PIN);                                 // criando a instância
decode_results results;                                  // declarando os resultados
bool LED5, LED6, LED7  = false;                          // estado dos LEDs
int atraso = 250;                                        // atraso após ligar LED

void setup()
{
  pinMode(7, OUTPUT);                                    // LED vermelho no pino D07
  pinMode(6, OUTPUT);                                    // LED azul no pino D06
  pinMode(5, OUTPUT);                                    // LED verde no pino D05
  irrecv.enableIRIn();                                   // Inicializa a recepção de
códigos
}

void loop()
{
  results.value = 0;                                     // zera os registradores
  if (irrecv.decode(&results))                           // se algum código for recebido
  {
    irrecv.resume();                                     // reinicializa o receptor
  }
  if (results.value == 0xFFB04F)                         // pressione tecla 3 para
controlar LED vermelho (D07)
  {
    LED7 = !LED7;                                        // alterna o estado do LED D07
    digitalWrite(7, LED7);                               // acende ou apaga LED vermelho
(D07)
    delay(atraso);                                       // atraso de 250 ms
  }

  if (results.value == 0xFF9867)                         // pressione tecla 2 para
controlar LED azul (D06)
  {
    LED6 = !LED6;                                        // alterna o estado do LED D06
    digitalWrite(6, LED6);                               // acende ou apaga LED vermelho
(D07)
    delay(atraso);                                       // atraso de 250 ms
  }

  if (results.value == 0xFF6897)                         // pressione tecla 1 para
controlar LED verde (D05)
  {
    LED5 = !LED5;                                        // alterna o estado do LED D05
    digitalWrite(5, LED5);                               // acende ou apaga LED verde (D05)
    delay(atraso);                                       // atraso de 250 ms
  }
}
```

Exemplo 13 – Módulo Sensor de Chuva e Sensor de Umidade do Solo

Sensor de umidade do solo

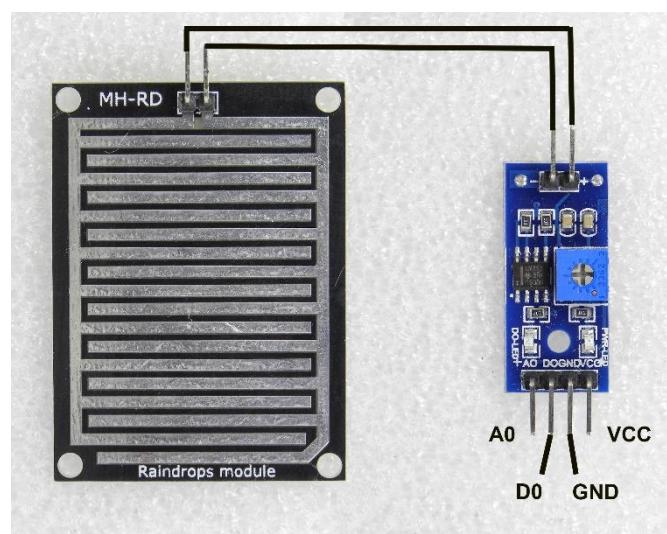
O sensor de umidade do solo mede basicamente a resistividade da terra. Se a terra estiver molhada, menor será a resistividade. E se a terra estiver seca, a resistividade será maior. O dispositivo que deverá ser enterrado no solo, consiste de duas superfícies metalizadas e isoladas uma da outra. A terra molhada sobre essas superfícies, permitirá a passagem de corrente. Essa corrente passando pela resistência do solo desenvolverá uma diferença de potencial (tensão), que será medida pelo conversor ADC do Arduino (portas analógicas).

Quanto maior a tensão medida pelo Arduino, menor será a umidade do solo. No caso do solo úmido, a tensão deverá ser mais baixa.

A conexão entre o sensor YL-69 e o módulo comparador deverá ser feita com fios. Não existe polaridade nesses fios. Recomendo a soldagem desses fios e o isolamento com tubos retráteis (quando aquecidos). Como ficarão sujeitos à umidade e água, as conexões precisam ficar protegidas e isoladas. Acho que um pouco de cola quente, pode ser útil também para proteger as partes sujeitas à corrosão (conexões). O módulo comparador deverá ficar distante das plantas e protegido contra água e umidade.

Esse módulo comparador tem um potenciômetro que serve para ajuste da sensibilidade do sensor. Tem uma saída analógica (A0) e uma saída digital (D0). A saída analógica será usada para a medição da tensão no sensor de umidade. Existe um resistor de 10K ohms que conecta um dos pinos do sensor ao 5V. O outro pino do sensor está conectado ao pino terra (GND).

Módulo Sensor de umidade do solo

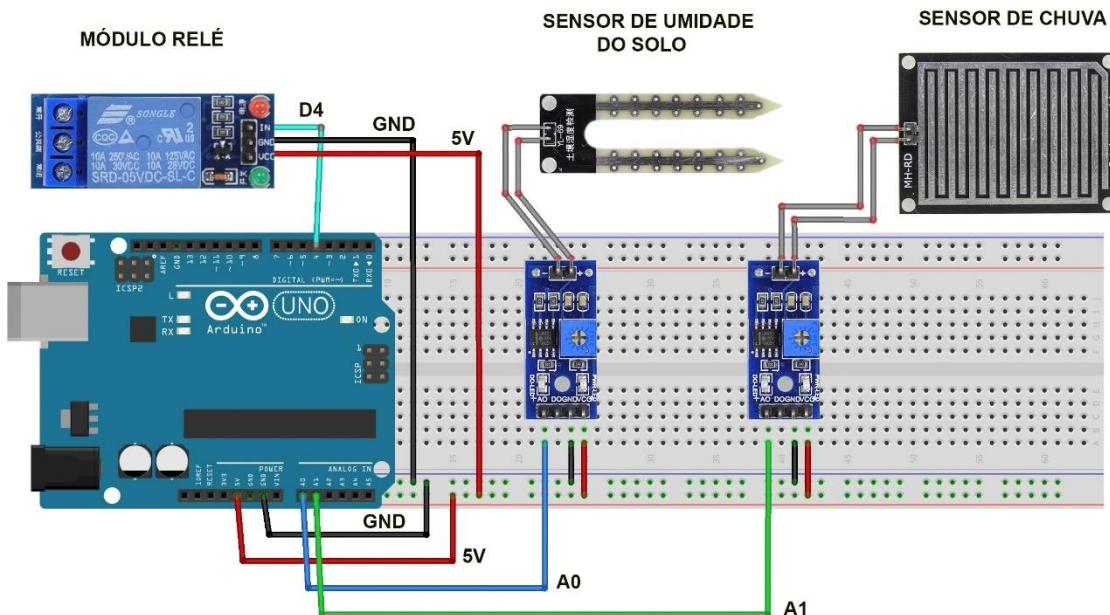
Já a saída digital corresponde à uma comparação de um determinado valor de tensão, feita por um amplificador operacional e ajustada pelo potenciômetro. Através de testes, você conseguirá determinar um valor de tensão (limite de umidade). Se variar para um valor menor do que o determinado, a saída digital (D0) mudará de estado de HIGH para LOW e o led D0 verde acenderá. Normalmente os potenciômetros já vem ajustados para a metade da tensão de alimentação do sensor.

O pino GND do módulo deverá ser conectado no pino terra do Arduino. E o pino VCC deverá ser conectado ao 5V, também do Arduino. Ao energizar o módulo, o led D1 verde irá acender.

Sensor de chuva

O Sensor de chuva tem o mesmo princípio de funcionamento do Sensor de umidade do solo. A diferença está na maior quantidade de linhas de detecção das gotas de chuva. Quando as gotas caem sobre a superfície do Sensor MH-RD, a resistividade do circuito diminui, portanto a tensão também. Quando a tensão estiver alta – sem chuva, tensão baixa – chuva!

O mesmo Módulo comparador poderá ser usado para o Sensor de chuva. Mas como serão montados os dois sensores em um mesmo Arduino, será necessário um módulo comparador para cada sensor.

Recomendo também a soldagem dos fios de conexão do sensor e o isolamento com tubos retráteis (quando aquecidos). Como ficarão sujeitos à umidade e água, as conexões precisam ficar protegidas e isoladas.

Módulo Sensor de chuva

Lista de Materiais:

Para este exemplo você vai usar os seguintes componentes:

- Arduino UNO,
- Protoboard,
- Jumpers de ligação,
- Sensor de umidade do solo,
- Sensor de chuva,
- Módulo Relé – 2 canais.

Montagem dos Sensores com Arduino

Essa é a montagem do Sensor de umidade do solo e Sensor de chuva com o Arduino. Acrescentei um módulo de relé para acionamento de uma pequena bomba d'água para irrigação das plantas. Use o contato normalmente aberto do relé para controlar o circuito da bomba d'água.

Obs: a bomba d'agua não consta no Kit Maker. No diagrama abaixo consta o Módulo com um Relé. Com esse Kit, use o Módulo Relé Dois canais.

O Sensor de umidade do solo deverá ser enterrado na terra do vaso ou do jardim. O sensor de chuva poderá ficar em um local próximo, mas deverá ficar inclinado num ângulo de 45 graus aproximadamente. Para que as gotas possam escorrer e secar, depois que a chuva parar. Os fios de conexões dos sensores deverão ser protegidos contra a água e umidade, como já havia informado.

Como nessa montagem, não usaremos as portas digitais, não haverá necessidade de ajustes dos potenciômetros. O ajuste de detecção de umidade e de chuva deverá ser feito através da configuração de parâmetros no Sketch (programa). Através da tentativa e erro poderá fazer um ajuste mais preciso, mas as configurações já estão definidas para a metade da tensão de alimentação dos sensores.

Código Arduino – Arduino Sensor de umidade e Sensor de chuva

O programa na próxima página, monitora o Sensor de umidade do solo e o sensor de chuva a cada 5 segundos. Se o solo estiver seco e não estiver chovendo, o Arduino acionará o relé da bomba para a irrigação. Começa um contador de tempo e a cada seis horas se ambos sensores estiverem secos, a bomba d'água será novamente acionada.

A duração de tempo de acionamento da bomba d'agua, deverá ser determinada através de testes. O tempo vai depender da quantidade de água que for bombeada para a planta. Recomendo o uso de uma pequena bomba d'água (talvez uma bomba de aquário).

Sugestões interessantes

Se você quiser implementar um controle melhor das condições climáticas do seu jardim, sugiro que acrescente também uma monitoração da umidade e temperatura do ar. Veja no exemplo:

Exemplo 8 - Sensor De Umidade e Temperatura DHT11

Para programar o horário de irrigação das plantas, poderá acrescentar um relógio RTC. Imagine irrigar somente na parte da manhã ou da noite, quando o clima é mais ameno! Faça o seu sketch usando o exemplo:

Exemplo XX - RTC – Real Time Clock DS 1307

```
// Exemplo 13 - Arduino Sensor de umidade do solo e Sensor de chuva
// Apostila Eletrogate - KIT MAKER
// Gustavo Murta

int sensorUmidadeSolo = A0;          // Sensor de umidade do solo pino A0 conectado no A0 do Arduino
int sensorChuva = A1;                // Sensor de chuva pino A0 conectado no A1 do Arduino
int portaRele = 4;                   // porta de controle do relé conectada no D4 do Arduino
int valorLimiteUmidade = 500;        // valor da tensão de comparação do sensor / valor máximo = 1024
int valorLimiteChuva = 500;          // valor da tensão de comparação do sensor máximo = 1024
bool chuva;                          // condição de chuva
bool soloUmido;                      // condição de solo úmido

void setup() {
  pinMode(sensorUmidadeSolo, INPUT);   // Sensor de umidade do solo - porta A0 é entrada
  pinMode(sensorChuva, INPUT);         // Sensor de chuva - porta A1 é entrada
  pinMode(portaRele, OUTPUT);          // Porta de controle do Relé - D4 é saída
  digitalWrite(portaRele, HIGH);       // Mantenha relé desligado
  Serial.begin(9600);                  // Monitor console 9600 Bps
}
void SensorDeChuva ()
{
  int valorSensorChuva = analogRead(sensorChuva);            // fazendo a leitura do Sensor de chuva
  Serial.print(" Sensor de chuva = ");                       // imprime mensagem
  Serial.print(valorSensorChuva);                            // imprime o valor do sensor de chuva
  if (valorSensorChuva < valorLimiteChuva)                   // se o valor for menor do que o limite
  {
    Serial.println("  => Esta chovendo");                    // imprime a mensagem
    chuva = 1 ;                                              // esta chovendo
  }
  else                                                       // senão
  {
    Serial.println("  => O tempo esta seco");                // imprime a mensagem
    chuva = 0 ;                                              // não esta chovendo
  }
}
void SensorDeUmidade ()
{
  int valorSensorUmidadeSolo = analogRead(sensorUmidadeSolo);   // leitura do Sensor de umidade do solo
  Serial.print(" Sensor de umidade do solo = ");               // imprime mensagem
  Serial.print(valorSensorUmidadeSolo);                        // valor do sensor de umidade do solo
  if (valorSensorUmidadeSolo < valorLimiteUmidade)             // se o valor for menor do que o limite
  {
    Serial.println("  => O solo esta úmido");                  // imprime a mensagem
    soloUmido = 1 ;                                            // solo esta úmido
  }
  else                                                         // senão
  {
    Serial.println("  => O solo esta seco");                   // imprime a mensagem
    soloUmido = 0 ;                                            // solo não esta úmido
  }
}
void ControleDoRele()                            // ajuste o tempo de acionamento da bomba
{
  digitalWrite(portaRele, HIGH);                 // relé desligado
  digitalWrite(portaRele, LOW);                  // relé ligado
  Serial.println(" Relé acionado ");             // imprime a mensagem
  delay (1000);                                  // tempo de acionamento da bomba dágua = 1 segundo
  digitalWrite(portaRele, HIGH);                 // relé desligado
}
void loop()
{
  for (int i = 1; i < 4321; i++)             // contagem de 1 a 4320  ( 4320 x 5 segundos = 6 horas)
  {
    SensorDeChuva ();                        // faz medição do sensor de chuva
    SensorDeUmidade ();                      // faz medição do sensor de umidade do solo
    if (chuva == 0 && soloUmido == 0 && i == 1) ControleDoRele();  // aciona a bomba dágua a cada 6
horas se tempo e solo estiverem secos
    Serial.print(i*5);                                 // imprime tempo em segundos
    Serial.println(" segundos"); Serial.println();     // imprime mensagem e uma linha
    delay(5000);                                       // atraso de 5 segundos
  }
}
```

Referências:

- http://blog.eletrogate.com/guia-do-sensor-de-umidade-do-solo-e-sensor-de-chuva/
- https://randomnerdtutorials.com/guide-for-rain-sensor-fc-37-or-yl-83-with-arduino/
- https://randomnerdtutorials.com/guide-for-soil-moisture-sensor-yl-69-or-hl-69-with-the-arduino/
- https://learn.sparkfun.com/tutorials/soil-moisture-sensor-hookup-guide

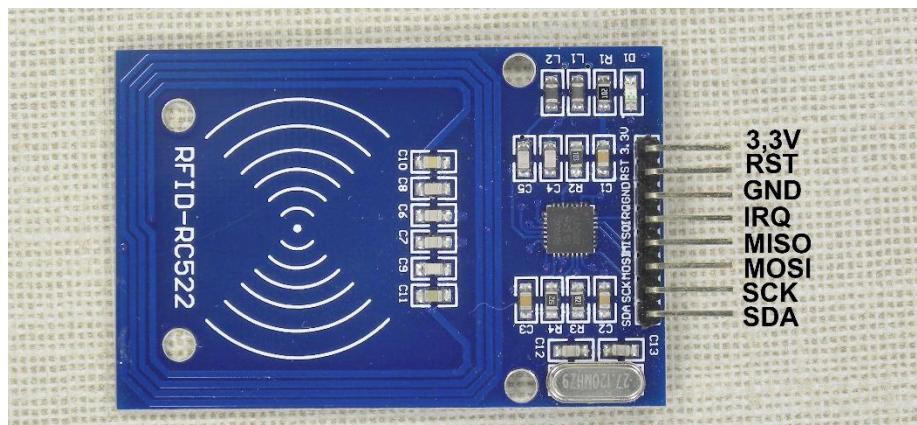
Exemplo 14 – Módulo RFID Arduino

RFID significa Identificação por Rádio Frequência (Radio Frequency Identification). É um sistema de identificação que usa as ondas eletromagnéticas para transferir dados de um dispositivo para outro à uma curta distância. O RFID é usado principalmente em controle de acesso à ambientes restritos, como na portaria de uma empresa para liberar a roleta de entrada para funcionários.

Os dispositivos que contém as informações de acesso são as etiquetas (tags), que possuem um microcontrolador bem pequenino e uma antena. O RFID funciona usando indução eletromagnética: um leitor emite uma pequena corrente elétrica que cria um campo eletromagnético. Esse campo é recebido por uma bobina semelhante na etiqueta, onde é transformada novamente em impulsos elétricos para transmitir e receber dados, como informações de identificação. O interessante é que a energia usada para ativar o chip interno da etiqueta é transmitida pela própria rádio frequência e é captada pela antena.

O Módulo RFID utiliza as ondas de rádio na frequência de 13,56 MHz. Mas existem também outros tipos de módulos com outras frequências de portadora.

Cada etiqueta tem uma identificação única (UID), para que nunca haja duplicidade de identificação. Esses são alguns tipos de dispositivos RFID, podendo ser chaveiros, cartões, pulseiras, etc.

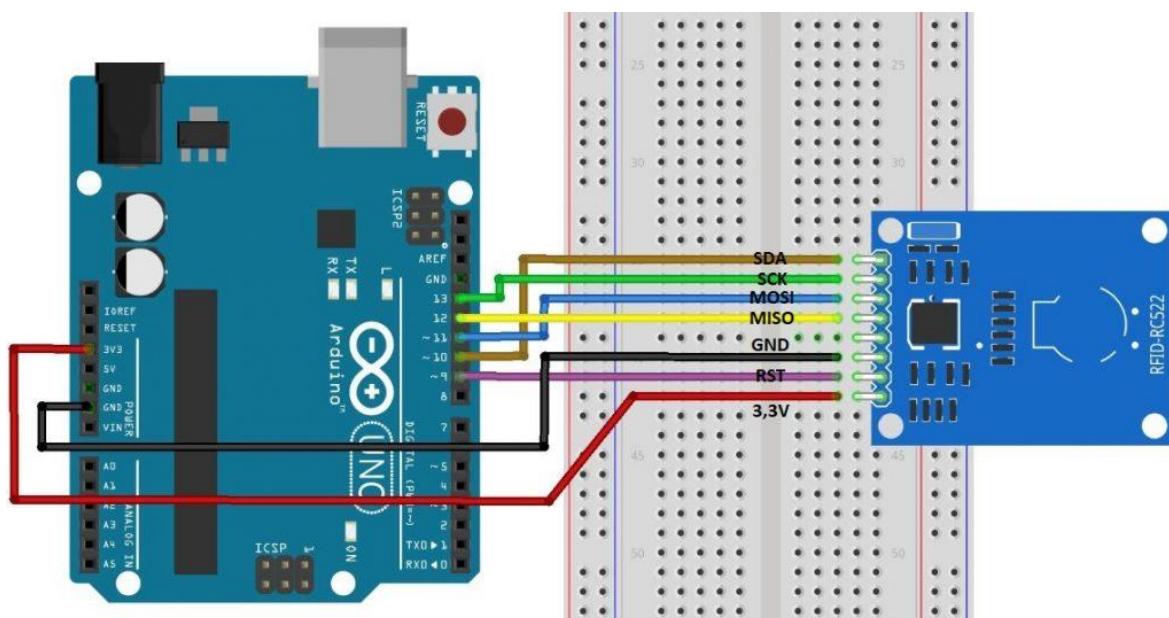
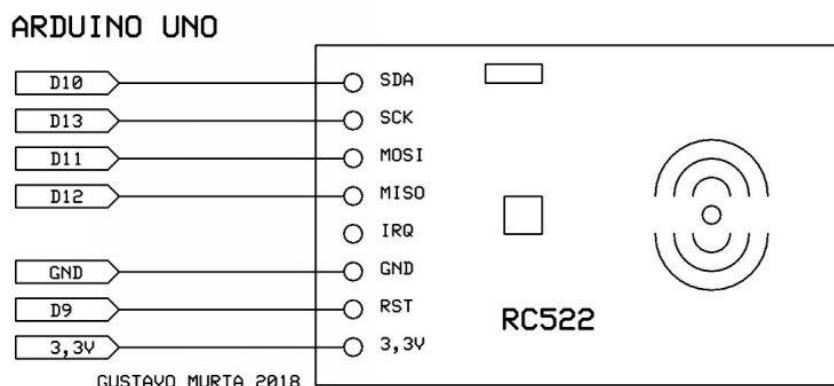
Módulo RFID MFRC522:

O módulo RFID do **Kit Maker** usa o Chip MFRC522 que é um microcontrolador que permite a transmissão e recepção dos dados das etiquetas RFID. A comunicação do chip com outros microcontroladores como o Arduino, é realizada através da interface SPI.

O protocolo de comunicação RFID usado nesse chip se chama MIFARE. A distância típica que a etiqueta deve estar do módulo é de apenas 5 cm. Quanto mais próxima ela estiver, melhor será o funcionamento.

Datasheet (folha de especificações) do chip MFRC522:
https://www.nxp.com/docs/en/data-sheet/MFRC522.pdf

A alimentação do módulo RFID é de 3,3V. As portas de controle e de comunicação SPI do Chip podem ser conectadas diretamente nas portas do Arduino (5V). A tensão das portas fica no limite, mas funcionam normalmente. Por isso não são necessários conversores de níveis de tensão.

Módulo RFID MFRC522 – pinagem

Lista de Materiais:
Para este exemplo você usará os seguintes componentes:

- Arduino UNO,
- Protoboard,
- Jumpers de ligação,
- Módulo RFID MFRC522.

Montagem do Modulo RFID MFRC522 com Arduino:

Muita atenção! O pino de alimentação do módulo deve ser conectado no pino 3,3V do Arduino. Não conecte no pino 5V, pois danificará o módulo.

Essas são as ligações que devem ser feitas com os jumpers:

Instalando a Biblioteca MFRC522:

A Biblioteca MFRC522 será usada nesse exemplo. Para instalar a nova biblioteca na IDE Arduino, clique em:

Sketch > Incluir Biblioteca > Gerenciar Bibliotecas

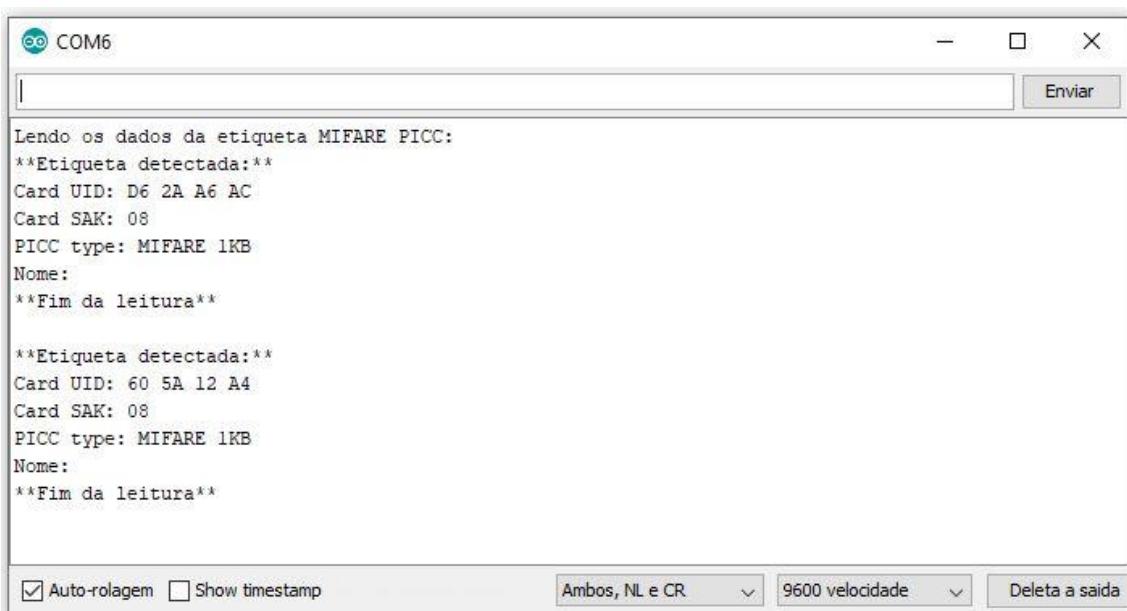
Após abrir a janela do Gerenciador de Biblioteca, refine a busca digitando MFRC522. Na biblioteca MFRC522, clique em More Info e depois em Instalar. Após alguns segundos, ela será automaticamente instalada. Lembre-se que o seu computador precisa estar conectado na internet, para poder baixar a biblioteca. Após a instalação da Biblioteca, é necessário que feche e abra novamente o programa Arduino IDE.

Se quiser estudar e testar os exemplos, esse é o link da Biblioteca MFRC522:
https://github.com/miguelbalboa/rfid

Para simplificar a montagem e o entendimento de como o Módulo RFID funciona, adaptei um exemplo da Biblioteca MFRC522 - **rfid_read_personal_data.ino**. Nesse código, quando aproximar um chaveiro ou cartão RFID do módulo, ele será lido e algumas informações do mesmo serão mostradas na console serial da IDE.

Para montar uma aplicação prática, como controle de acesso com RFID, sugiro que baseie se nos tutoriais que estão nos links de referência, logo abaixo.

Código Arduino – Módulo RFID:

```
// Exemplo 14 - Modulo RFID com Arduino
// Apostila Eletrogate - KIT MAKER
// Gustavo Murta
// https://github.com/miguelbalboa/rfid/tree/master/examples/rfid_read_personal_data

#include <SPI.h>                                    // bilblioteca SPI
#include <MFRC522.h>                                // biblioteca MFRC522
#define RST_PIN 9                                   // pino Reset = Arduino D9
#define SS_PIN  10                                  // pino SS(SPI) = Arduino D10
MFRC522 mfrc522(SS_PIN, RST_PIN);                   // cria a instancia MFRC522

void setup()
{
  Serial.begin(9600);                                              // console serial 9600 bps
  SPI.begin();                                                     // inicializa interface SPI
  mfrc522.PCD_Init();                                              // inicializa o modulo MFRC522
  Serial.println(F("Lendo os dados da etiqueta MIFARE PICC:"));    // imprime mensagem console serial
}

void loop()
{
  MFRC522::MIFARE_Key key;                                // preparando as chaves
  for (byte i = 0; i < 6; i++) key.keyByte[i] = 0xFF;     // montando as chaves

  byte block;                                             // algumas variaveis necessarias
  byte len;
  MFRC522::StatusCode status;                             // lendo o estado das etiquetas
  if ( ! mfrc522.PICC_IsNewCardPresent())                 // se nao houver etiqueta
  {
    return;
  }
  if ( ! mfrc522.PICC_ReadCardSerial())                   // selecione uma das etiquetas
  {
    return;
  }
// Continua na próxima página
```

```
  Serial.println(F("**Etiqueta detectada:**"));          // imprime mensagem
  mfrc522.PICC_DumpDetailsToSerial(&(mfrc522.uid));      // mostra informacoes da etiqueta
  Serial.print(F("Nome: "));                             // imprime mensagem

  byte buffer1[18];
  block = 4;
  len = 18;
  status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, 4, &key, &(mfrc522.uid));
  if (status != MFRC522::STATUS_OK)                       // se falhar na leitura
  {
    Serial.print(F("Falha na autenticacao: "));           // imprime mensagem
    Serial.println(mfrc522.GetStatusCodeName(status));    // imprime o erro
    return;
  }
  status = mfrc522.MIFARE_Read(block, buffer1, &len);
  if (status != MFRC522::STATUS_OK)                       // se falhar na leitura
  {
    Serial.print(F("Falha na leitura: "));                // imprime mensagem
    Serial.println(mfrc522.GetStatusCodeName(status));    // imprime o estado
    return;
  }
  for (uint8_t i = 0; i < 16; i++)                        // para 15 ponteiros
  {
    if (buffer1[i] != 32)                                 // se o buffer for diferente de 32
    {
      Serial.write(buffer1[i]);                           // imprime o buffer
    }
  }
  Serial.print(" ");
  Serial.println(F("\n**Fim da leitura**\n"));            // imprime mensagem
  delay(1000);                                            // atraso de 1 segundo
  mfrc522.PICC_HaltA();                                   // para a leitura do RFID
  mfrc522.PCD_StopCrypto1();
}
```

Janela da console serial da IDE Arduino – um cartão e um chaveiro foram lidos.

Referências:

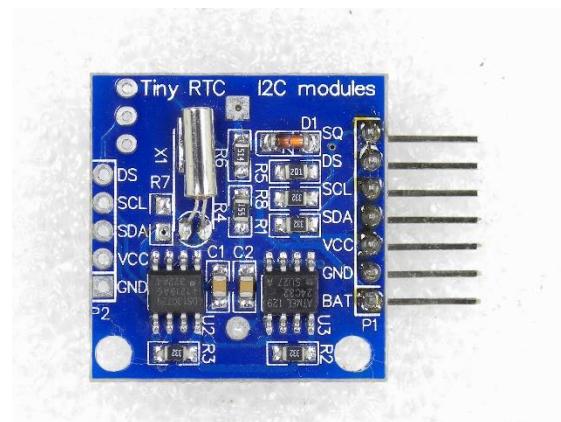
- https://blog.eletrogate.com/guia-basico-da-nfc-para-arduino/
- https://blog.eletrogate.com/kit-rfid-com-arduino-sistema-de-controle-de-acesso/
- https://randomnerdtutorials.com/security-access-using-mfrc522-rfid-reader-with-arduino/
- https://howtomechatronics.com/tutorials/arduino/rfid-works-make-arduino-based-rfid-door-lock/

Exemplo 15 – Relógio RTC – DS1307

Módulo RTC DS1307:

Esse módulo serve como relógio para projetos com Arduino que precisam do controle do tempo. Pode ser usado como alarme despertador, controle automatizado de operações temporizadas, e claro como um relógio comum com horas e minutos. O módulo tem o chip DS1307 que é o relógio, uma EEPROM 24C32, um cristal de 32,768 KHz e uma pilha CR2032.

O chip DS1307 RTC (Real-Time Clock) é um relógio / calendário de baixa potência com codificação binária completa (BCD) mais 56 bytes de NV SRAM (memória estática RAM não-volátil). Os endereços e os dados são transferidos em série por meio do barramento bidirecional da interface I2C. O relógio / calendário fornece informações sobre segundos, minutos, horas, dia, data, mês e ano. O relógio funciona no formato de 24 ou 12 horas com o indicador AM / PM. O DS1307 possui um circuito de detecção de falha de energia que alterna automaticamente a alimentação para a pilha CR2032 (3V), na ausência de energia da fonte. A contagem de tempo do relógio é baseada no oscilador do cristal de 32,768 KHz.

DS1307 – folha de especificações (datasheet):
https://datasheets.maximintegrated.com/en/ds/DS1307.pdf

A memória EEPROM 24C32 permite o armazenamento de até 4096 K Bytes e ela é não volátil, isto é, mesmo sem energia ela armazena os dados gravados. Para a comunicação com o microcontrolador ela também usa a interface I2C.

AT24C32 – folha de especificações (datasheet):
http://ww1.microchip.com/downloads/en/devicedoc/doc0336.pdf

Na placa do circuito do módulo, existe um espaço onde um sensor de temperatura DS18B20 poderá ser soldado. Esse sensor é opcional e não vem com o KIT.

Módulo RTC – DS1307

Pinagem do Módulo DS1307:

- SQ – Onda quadrada (normalmente não usado)
- DS – Sensor DS18B20 (opcional)
- SCL – Serial Clock (I2C Interface)
- SDA – Serial Data (I2C interface)
- VCC – Alimentação 5V (mínimo: 4,5V)
- GND – Terra
- BAT – Tensão da bateria (aproximadamente 3V)

A alimentação do módulo RTC DS1307 deve ser de 5V. A tensão mínima de alimentação do chip DS1307 é de 4,5V. Portanto esse módulo não pode ser alimentado com 3,3V.

Os pinos de comunicação I2C do módulo já tem os resistores de pullup. Por isso não é necessário acrescentar esses resistores. Os pinos podem ser conectados diretamente no Arduino Uno.

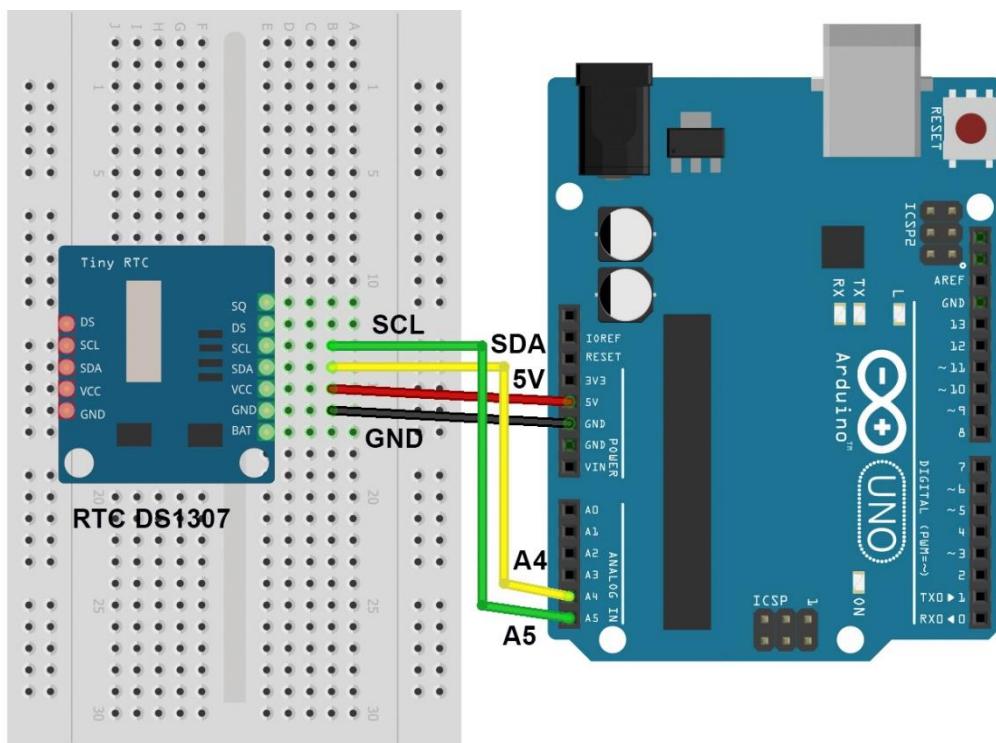
A tensão da bateria (pilha de Lítio) deverá variar com o uso. Quando a pilha está nova, a tensão é de 3,2V aproximadamente.

Lista de Materiais:

Para este exemplo você usará os seguintes componentes:

- Arduino UNO,
- Protoboard,
- Jumpers de ligação,
- Módulo RTC DS1307.

Montagem do Modulo RTC DS1307 com Arduino:

Faça as ligações de acordo:

SCL RTC => A5 Arduino

SDA RTC => A4 Arduino

VCC RTC => 5V Arduino

GND RTC => GND Arduino

Instalando a Biblioteca RTClib:

A Biblioteca RTClib será usada nesse exemplo. Para instalar a nova biblioteca na IDE Arduino, clique em:

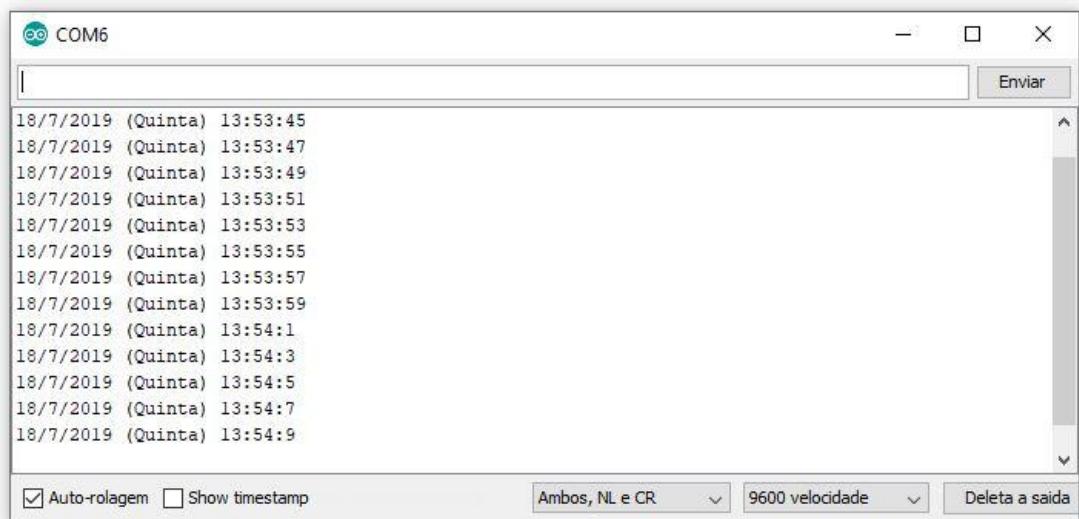
Sketch > Incluir Biblioteca > Gerenciar Bibliotecas

Após abrir a janela do Gerenciador de Biblioteca, refine a busca digitando RTClib. Na biblioteca RTClib, clique em More Info e depois em Instalar. Após alguns segundos, ela será automaticamente instalada. Lembre-se que o seu computador precisa estar conectado na internet, para poder baixar a biblioteca. Após a instalação da Biblioteca, é necessário que feche e abra novamente o programa Arduino IDE.

Esse é o link da Biblioteca RTClib:

https://github.com/adafruit/RTClib

Código Arduino – Módulo RTC DS1307:

```
// Exemplo 15 – Modulo RTC DS1307 com Arduino
// Apostila Eletrogate - KIT MAKER
// Gustavo Murta
// https://github.com/adafruit/RTClib/blob/master/examples/ds1307/ds1307.ino

#include "RTClib.h"                        // biblioteca RTClib

RTC_DS1307 rtc;                            // criando a instância RTC

char diasDaSemana[7][12] = {"Domingo", "Segunda", "Terca", "Quarta", "Quinta", "Sexta", "Sabado"};

void setup ()
{
  Serial.begin(9600);                                   // monitor da console serial 9600 bps
  if (! rtc.begin())                                    // se o RTC não foi inicializado
  {
    Serial.println("RTC nao pode ser encontrado!");     // imprime mensagem
    while (1);
  }

  if (! rtc.isrunning())                                // se o RTC não estiver rodando
  {
    Serial.println("RTC nao esta funcionando!");        // imprime mensagem
    rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));     // ajusta relógio com o relógio do seu PC
  }
}

void loop ()
{
  DateTime now = rtc.now();                            // faz a leitura dos dados do RTC
  Serial.print(now.day(), DEC);                        // imprime o dia do mês
  Serial.print('/');
  Serial.print(now.month(), DEC);                      // imprime o mês
  Serial.print('/');
  Serial.print(now.year(), DEC);                       // imprime o ano
  Serial.print(" (");
  Serial.print(diasDaSemana[now.dayOfTheWeek()]);      // imprime o dia da semana
  Serial.print(") ");
  Serial.print(now.hour(), DEC);                       // imprime horas
  Serial.print(':');
  Serial.print(now.minute(), DEC);                     // imprime minutos
  Serial.print(':');
  Serial.print(now.second(), DEC);                     // imprime segundos
  Serial.println();                                    // imprime uma linha
  delay(2000);                                         // atraso de 2 segundos
}
```

Esse programa foi traduzido e adaptado do exemplo DS1307 da biblioteca RTClib. Ele acerta a data e as horas com o relógio do seu PC. E fica imprimindo a data e as horas, a cada 2 segundos. Esse exemplo é para você entender basicamente como o relógio RTC pode ser usado.

Sabendo-se que o módulo tem uma memória EEPROM e pode ainda ter um sensor de temperatura (opcional), você poderá criar inúmeras aplicações bem interessantes.

Recomendo que estude as referências abaixo para poder desenvolver novos projetos e fazer novas montagens.

Veja a data e o horário na console serial (9600 Bps):

Referências:

- https://blog.eletrogate.com/rtc-real-time-clock-ds1302-1307-e-3231/
- https://learn.adafruit.com/ds1307-real-time-clock-breakout-board-kit/overview
- https://wiki.dfrobot.com/Real_Time_Clock_Module__DS1307___SKU_DFR0151_
- https://howtomechatronics.com/projects/arduino-touch-screen-music-player-alarm-clock-project/

Considerações finais

Essa apostila tem por objetivo apresentar alguns exemplos básicos sobre como utilizar os componentes do Kit Arduino MAKER, a partir dos quais você pode combinar e fazer projetos mais elaborados por sua própria conta.

Nas seções de referências de cada exemplo e nas referências finais, também tentamos indicar boas fontes de conteúdo objetivo e com projetos interessantes. Sobre isso ponto, que consideramos fundamental, gostaríamos de destacar algumas fontes de conhecimento que se destacam por sua qualidade.

O fórum oficial Arduino possui muitas discussões e exemplos muito bons. A comunidade de desenvolvedores é bastante ativa e certamente pode te ajudar em seus projetos. No Project Hub poderá encontrar milhares de projetos com Arduino.

- Fórum oficial Arduino: https://forum.arduino.cc/
- Project Hub Arduino: https://create.arduino.cc/projecthub

O Instructables é a ótima referência do mundo maker atual. Pessoas que buscam construir suas próprias coisas e projetos encontram referências e compartilham suas experiências no site.

- Instructables: https://www.instructables.com/

O Maker pro é outro site referência no mundo em relação aos projetos com Arduino. Há uma infinidade de projetos, todos bem explicados e com bom conteúdo.

- Maker pro: https://maker.pro/projects/arduino

Em relação à eletrônica, teoria de circuitos e componentes eletrônicos em geral, deixamos alguns livros essenciais na seção finais de referências. O leitor que quer se aprofundar no mundo da eletrônica certamente precisará de um livro basilar e de bons conhecimentos em circuitos eletroeletrônicos.
No mais, esperamos que essa apostila seja apenas o início de vários outros projetos e, quem sabe, a adoção de Kits mais avançados, como os de **Robótica** e **Arduino Advanced**.
Qualquer dúvida, sugestão, correção ou crítica a esse material, fique à vontade para relatar em nosso blog oficial: http://blog.eletrogate.com/

Referências gerais

1. Fundamentos de Circuitos Elétricos. Charles K. Alexander; Matthew N. O. Sadiku. Editora McGraw-Hill.
2. Circuitos elétricos. James W. Nilsson, Susan A. Riedel. Editora: Pearson; Edição: 8.
3. Microeletrônica - 5ª Ed. - Volume Único (Cód: 1970232). Sedra,Adel S. Editora Pearson.
4. Fundamentals of Microelectronics. Behzad Razavi. Editora John Wiley & Sons; Edição: 2nd Edition (24 de dezembro de 2012).

Abraços e até a próxima!

Vitor Vidal
Engenheiro eletricista, mestrando em eng. elétrica e apaixonado por eletrônica, literatura, tecnologia e ciência. Divide o tempo entre pesquisas na área de sistemas de controle, desenvolvimento de projetos eletrônicos e sua estante de livros.

Partes editadas por Vitor Vidal:

Parte I - Revisão de circuitos elétricos e componentes básicos
Parte II - Seção de Exemplos Práticos - Exemplos 1 ao 6 e 10.

Gustavo Murta

Técnico em eletrônica, formado em Curso superior de TPD, com pós-graduado em Marketing. Trabalhou por muitos anos na IBM na área de manutenção de computadores de grande porte. Aposentou-se, podendo curtir o que mais gosta: estudar e ensinar Tecnologia. Hobbista em eletrônica desde 1976. Gosta muito de Fotografia e Observação de aves.

Partes editadas por Gustavo Murta:

Revisão de todas as partes editadas por Vitor Vidal.
Exemplo 7 - Módulo Relé 2 Canais
Exemplo 8 - Sensor De Umidade e Temperatura DHT11
Exemplo 9 - Display LCD 16x2
Exemplo 11 - Display de 7 segmentos + Teclado de membrana
Exemplo 12 - Controle Remoto Ir + Receptor Universal Ir
Exemplo 13 - Módulo Sensor de Chuva e Sensor de Umidade do Solo
Exemplo 14 – Módulo RFID Arduino
Exemplo 15 – Módulo RTC DS1307

Versão 1.0 – julho de 2019

APOSTILA KIT

ARDUINO MAKER

Esta apostila acompanha o **Kit ARDUINO MAKER** da Eletrogate, e contém conteúdos relacionados a todos os componentes do Kit.

WWW.ELETROGATE.COM